COMPARISON OF TRADITIONAL VERSUS CUBESAT REMOTE SENSING: A MODEL-BASED SYSTEMS ENGINEERING APPROACH

THESIS

Daniel L. Cipera, Captain, USAF

AFIT-ENV-MS-18-M-187

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENV-MS-18-M-187

COMPARISON OF TRADITIONAL VERSUS CUBESAT REMOTE SENSING: A MODEL-BASED SYSTEMS ENGINEERING APPROACH

THESIS

Presented to the Faculty

Department of Systems Engineering and Management

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Systems Engineering

Daniel L. Cipera, BS

Captain, USAF

March 2018

AFIT-ENV-MS-18-M-187

COMPARISON OF TRADITIONAL VERSUS CUBESAT REMOTE SENSING: A MODEL-BASED SYSTEMS ENGINEERING APPROACH

Daniel L. Cipera, BS

Captain, USAF

Committee Membership:

Dr. D. Jacques
Chair

Dr. T. Ford
Member

Mr. D. Meyer
Member

AFIT-ENV-MS-18-M-187

Abstract

This thesis compares the ability of both traditional and CubeSat remote sensing architectures to fulfill a set of mission requirements for a remote sensing scenario. Mission requirements originating from a hurricane disaster response scenario are developed to derive a set of system requirements. Using a Model-based Systems Engineering approach, these system requirements are used to develop notional traditional and CubeSat architecture models. The technical performance of these architectures is analyzed using Systems Toolkit (STK); the results are compared against Measures of Effectiveness (MOEs) derived from the disaster response scenario. Additionally, systems engineering cost estimates are obtained for each satellite architecture using the Constructive Systems Engineering Cost Model (COSYSMO). The technical and cost comparisons between the traditional and CubeSat architectures are intended to inform future discussions relating to the benefits and limitations of using CubeSats to conduct operational missions.

Acknowledgments

I would like to express my sincere appreciation to my faculty advisors, Dr. Dave Jacques and Dr. Thomas Ford, along with Mr. David Meyer, for their guidance and support throughout the course of this thesis effort. Their insight and experience is certainly appreciated. I would also like to thank Dr. Ray Madachy of the Naval Postgraduate School for his assistance with the cost estimation portion of this thesis. Finally, my utmost gratitude to my wife, who somehow thought all this was a good idea, and my children, who had no choice in the matter.

Daniel L. Cipera

Table of Contents

List of Figures

Page

List of Tables

Page

COMPARISON OF TRADITIONAL VERSUS CUBESAT REMOTE SENSING: A MODEL-BASED SYSTEMS ENGINEERING APPROACH

I. Introduction

General Issue

Since the development of the CubeSat standard in 1999, CubeSats have become popular among the academic and scientific communities as educational tools and technology demonstration platforms (National Academy of Sciences, 2016: vii). However, much speculation has been given as to the possibility of using CubeSats as a cheaper alternative to larger, “traditional” spacecraft and satellite constellations for military and civilian operational missions. Smaller satellites sizes, with the associated reduction in material costs, complexity, and assembly timelines, suggest the possibility of accomplishing a given mission at a lower cost.

The validity of the above premise is dependent on the intended mission to be accomplished. Some missions, such as high-resolution radar imaging, have physical requirements that CubeSats may not meet; the electrical power required in this example is beyond the current capability of CubeSats to provide (Selva & Krejci, 2012). However, many relevant technologies have been demonstrated on CubeSats which may make certain mission sets possible. Specifically, practical Electro-Optical, or EO, sensors on CubeSats have been demonstrated on numerous academic, scientific, and commercial missions.

Current remote sensing platforms, such as DigitalGlobe’s WorldView-series EO imaging satellites, are remarkably capable; in disaster scenarios, where “high-resolution” imagery is considered 5 meters GSD or less (Hoque, Phinn, Roelfsema & Childs, 2017:345), the WorldView-series satellites can provide panchromatic images with resolution near 31 cm (DigitalGlobe, 2016). However, this performance comes with a significant cost; WorldView-4 cost an estimated $835 million to build and launch (Smith, 2012). For both government and commercial operators, constructing and operating a constellation with this level of capability is an expensive endeavor; in a fiscally constrained environment, a traditional architecture may have to sacrifice some degree of mission or performance in order to satisfy cost constraints.

Problem Statement

CubeSat architectures may provide a cheaper alternative to the expensive traditional systems described above; however, due to limitations in physical size, a CubeSat would not match mission performance compared to a traditional satellite. Given these limitations, it is not well understood how well a CubeSat architecture is able to perform operational missions typically executed by a traditional architecture, thus providing this cheaper alternative. Additionally, while CubeSats are logically thought of as cheaper than traditional satellites, few cost models are available for this design space to inform how much cheaper a CubeSat solution may be.

Research Objectives and Questions

This thesis has three main research objectives. First, develop appropriate medium-fidelity models of both traditional and CubeSat architectures for the purposes of architecture analysis and comparison. Second, use those models to investigate the suitability of using a CubeSat architecture to provide Essential Elements of Information

(EEIs) in a disaster response scenario. Third, use the same models as inputs to a systems engineering cost estimation model to determine the cost model's suitability towards satellite designs.

To pursue these research objectives, several research questions were investigated. These are:

1. Given a set of mission objectives and requirements, how well does a CubeSat and a traditional remote sensing architecture meet these requirements?
2. Are systems engineering cost models such as COSYSMO a valid and useful means of predicting and comparing systems engineering and program costs for traditional and CubeSat architectures?
3. What are the implications of using Model-Based Systems Engineering (MBSE) to answer questions one and two?

Methodology

The methodology for this thesis is derived from the Model-Based Systems Engineering (MBSE) approach. Mission requirements are derived from a hurricane disaster response scenario, along with Measures of Effectiveness (MOEs). Two architectures are modeled in SysML using those mission requirements and their derived system and functional requirements. STK is used to analyze the performance of each architecture; COSYSMO is used to provide systems engineering cost estimates. The STK results are compared to the MOEs to provide a clear picture of how well each architecture performs against the mission requirements.

Assumptions

Categorized by weight, satellites range along a scale from 100 gram “femto” satellites to “large” satellites weighing well over 1000 kg (Konecny, 2004). “Traditional satellites,” as defined in this thesis, refer to the large end of the scale, with the upper limit being the payload capacity of existing launch vehicles. Within Konecny’s scale, “nanosatellites” refers to spacecraft between 1 and 10 kg; this weight range corresponds with the “CubeSat” standard defined by Robert Twiggs at the Space System Development Laboratory, Stanford University (European Space Agency, n.d.). This thesis is limited to comparing the large/traditional and nano/CubeSat categories described here; while a middle range between 10 and 1000 kg does certainly exist, there is not as much historical basis for that range as compared to the traditional realm, nor is there a well-defined weight and volume standard for this range as there is for CubeSats.

In order to both simplify and scope this thesis, a remote sensing type and mission was decided on up front. Visible-spectrum EO was chosen because it is a mature technology with sensors in use on both traditional and CubeSat missions. Although EO is limited by cloud cover, cloud cover will not be specifically addressed, as this limitation applies to both traditional and CubeSat architectures.

To analyze performance, this thesis is limited to performance objectives traced back to a hurricane disaster response scenario. While this is a weather-based scenario, the use of EO to collect weather data is not considered. In order to keep this thesis openly distributable, military remote sensing applications are not considered.

The main focus of the architecture comparison is on functional performance and cost. Various “-ilities” such as flexibility and resiliency are not considered specifically; however future research could be done in this regard using the models developed here.

CubeSats have been discussed in the context of responsive spacelift or “launch on demand,” in which a capability is deployed when it is needed rather than in advance. Again, to limit scope, this is not discussed; it is assumed in this scenario that both architectures are deployed and operational prior to the beginning of the scenario.

II. Literature Review

Chapter Overview

This literature review contains three sections exploring three relevant topics. The first section provides context for using MBSE in space architecture performance and cost modeling. The second section summarizes research relating to CubeSat utility, operations and limitations. The third develops the background necessary to identify the Essential Elements of Information (EEIs) for a hurricane disaster scenario, along with other information necessary for development of this scenario.

Space Mission Architecture Modeling and Simulation

A methodology for assessing CubeSat architectures is discussed by Selva and Krejci; their method utilizes a genetic algorithm to optimize combinations of sensors and orbits to achieve some fraction of requirements from multiple inter-related mission sets. Once an optimized reference architecture is reached, its overall mission performance is modeled using STK. Additionally, Selva and Krejci propose a simple cost model incorporating recurring-and non-recurring bus, payload, and operation costs, along with launch costs (2013).

Thompson extends this methodology using an MBSE/Model-Based Conceptual Design (MBCD) approach, focusing on the analysis and optimization of disaggregated space architectures. While discussing MBCD, he notes that "integration of standardized systems engineering tools that are capable of integrating parametric cost models with functional and performance models could provide significant utility". Thompson concludes that MBSE and the Object-Oriented Systems Engineering Method (OOSEM) are effective methods of modeling disaggregated space systems (2015).

The usage of the Systems Modeling Language (SysML) and MBSE to model a CubeSat design is explored by Kaslow, Soremekun, Kim, and Spangelo (2014). Kaslow et al. develop a SysML CubeSat model using the MagicDraw modeling tool. Their model uses the executable functions of MagicDraw, along with STK, to analyze system performance. Kaslow et al. demonstrate the ability of a MagicDraw SysML model to perform component-level trade studies on their CubeSat design (2014).

While not specifically described as MBSE, Krueger, Selva, Smith and Keesee discuss the development and optimization of a smallsat imaging architecture for global crisis response using an "integrated model": a "parameterized representation of the spacecraft and ground stations that can be used to simulate competing system configurations" (2009). Much of their effort follows the standard system engineering process: identifying mission requirements, developing a concept of operations, and deriving system requirements. Subsequently, the authors use their integrated model to perform relevant trades and optimize constellation performance amongst the competing objectives of image resolution and responsiveness. Krueger et. al. used Matlab and STK to conduct these analyses (2009).

The usage of MBSE and SysML for systems engineering cost estimation is a relatively recent development. The COSYSMO systems engineering cost estimation tool was developed as part of a dissertation by Valerdi. COSYSMO is a parametric cost model that uses functional size, effort multipliers, and calibration and scale factors to estimate the system engineering effort needed to develop a given system. Functional size is estimated using “size drivers”: counts of system requirements, system interfaces, critical algorithms, and operational scenarios. Each individual requirement, interface, algorithm, and operational scenario is assessed to be easy, nominal, or difficult to implement; this rating becomes a multiplier as part of calculating functional size. Valerdi’s method was developed in the context of documents-based systems engineering, with size drivers counts derived from system specifications, interface control documents, and use cases (Valerdi, 2005).

The usage of MBSE to support COSYSMO analysis has been investigated by both Edwards (2016) and Pavalkis, Papke and Wang (2017). Using a water filtration system example, Edwards demonstrates that using SysML to model and count the design aspects that contribute to a system’s functional size is a practical approach. Edward’s mapping of COSYSMO size drivers to SysML diagrams is outlined in Table 1.

Table 1. Mapping of COSYSMO Size Drivers to SysML Diagrams. Modified from (Edwards, 2016)**.**

Size Driver	SysML Diagrams
Requirements	Requirements Diagram Package Diagram
Interfaces	Block Definition Diagram Internal Block Diagram
Algorithms	Block Definition Diagram Parametric Diagram
Operational Scenarios	Use Case Diagram

Edwards acknowledges that the water filtration example is a basic one, and that challenges may exist scaling this approach to larger systems (2016). More recently, Pavalkis, Papke and Wang have discussed in depth the practical details of taking a SysML model and using it for COSYSMO cost estimation, though they use a modified version of the COSYSMO model to account for development with reuse and development for reuse (2017).

CubeSat Utility, Operations and Limitations

Selva and Krejci empirically describe, using historic examples, how CubeSats could be used to fulfill scientific Earth Observation requirements as defined by the Committee on Earth Observing Satellites (CEOS). These authors also describe key limitations in each common subsystem, to include communications (data rates), ADACS (pointing accuracy), mass/size (limits aperture sizes, both optical and antenna), power (solar panel geometry limits power to about 1 Watt or so; this rules out any payloads, such as radar or LIDAR, that require much more power than a Watt), propulsion (limited capability in form of cold gas, vacuum arc thrusters), and thermal (mostly passive, though it might be possible to have an active battery heater) (Selva and Krejci, 2012).

While Selva and Krejci's discussion gives a starting point for understanding CubeSat limitations, advancements to overcome these limitations is ongoing. For example, Planet Labs has developed solar panels without cover glass for CubeSats, which they claim yields "significantly more power" for less cost and mass compared to previous solar panel designs. Additionally, Planet Labs claims to have identified a way to put an X-band transmitter on a nanosatellite, with data rates around 100 Mbps (Boshuizen, 2014:3). Wherever possible, the thesis uses Selva and Krejci's discussion of limitations to bound the design space, except in cases of known technology advancements such as Planet Lab's.

In her thesis, McKenney describes a CubeSat architecture for fulfilling the DoD's weather mission. McKenney's research demonstrates that a CubeSat architecture can fulfill the mission requirements of an operational weather mission, though in some cases only marginally. No detailed comparison to traditional satellites is made (McKenney, 2016: 70-72).

Essential Elements of Information (EEIs) for Hurricane Disaster Response

Immediately following a hurricane landfall, required information includes location, amount, rate, type, and percentage of areas and structures affected (Hoque et al., 2017:352). The utility of imagery in determining this information is highly dependent on sensor resolution. Note that sensor resolution can be an ambiguous term; at a basic level, it is "a limit on how small an object on the Earth's surface can be and still be 'seen' by a sensor as being separate from its surroundings" (Lillesand, Kiefer and Chipman, 2008:33). Much of the literature discussed below refers to resolution in more specific terms of "pixel size" and its corresponding Ground Sample Distance (GSD). GSD is defined technically as the instantaneous Field of View in one linear dimension for one pixel for a given sensor (Evans, Lange & Schmitz, 2014:184). For this literature review, the terms used are the same as what the authors used in their respective papers. After the literature review, Ground Sample Distance is discussed unless otherwise specifically stated.

Flooding can be monitored using medium resolution imagery. A pixel size[1] of 10 meters is sufficient for building identification and location, while discerning building

[1] Pixel size is Hoque et. al's terminology for pixel spacing; the term pixel size is used here to stay consistent with this source material.

damage requires a pixel size on the order of 1 m (Womble et. al., 2006:1). This conclusion is supported analytically by Battersby, Hodgson and Wang, who determined that 1.5 m is the threshold spatial resolution for an imagery analyst to assess residential building damage (2012:625). Similarly, Krueger et al. identify 1 m ground resolution, with a corresponding 0.5 m Ground Sample Distance, as sufficient requirements for imagery systems involved in disaster response (2009:5). Change detection products using moderate to very high resolution (less than 30 m; less than 10 m if looking at man-made objects) have been recommended for the disaster response phase (Hoque et al., 2017:352).

Utility of imagery is also dependent upon timeliness. Responding agencies need imagery within 72 hours of an event, ideally within 24 hours (Hodgson, Davis, Cheng & Miller, 2010:7). Krueger et al. specify a much shorter timeline of 4 hours from tasking to target access as requirement for a notional disaster response imagery system (2009:5); no detailed justification is made for this timeline.

Geographically, most hurricanes to strike the U.S. make landfall below 37° latitude; historically, 316 of the 342 hurricanes to strike the U.S. between 1850 and 2005 have hit at or below this latitude, with 247 making landfall at or below 31° latitude (Hodgson, Davis, Cheng & Miller, 2010:11). This information may help to determine the orbital inclination of architectures to optimize for coverage.

III. Methodology

Chapter Overview

The purpose of this chapter is to describe the methodologies by which the research questions are addressed. In order to develop candidate architectures for cost and performance analysis, an MBSE, or Model-based System Engineering, approach is used. Architecture models for analysis are created using the Object-Oriented Systems Engineering Method (OOSEM), with relevant views generated in SysML using the Cameo Systems Modeler tool. Each model is evaluated against performance MOEs using STK. Costs are compared using the COSYSMO cost model, using relevant aspects of the architectures models as input.

In order to make determinations about the benefits and limitations of traditional and CubeSat architectures for a given mission, measures to compare these architectures against must be developed. There are many missions for which CubeSats have potential; however, spaceborne imagery for remote sensing is a well-known and mature capability, making it particularly suitable as a basis for this analysis.

Choosing a Mission and Defining MOEs

Remote sensing platforms conduct several missions, including natural disaster response. Historically, satellite imagery has assisted in the response to earthquakes, floods, forest fires, and hurricanes. Any of these scenarios could have been used to derive MOEs and mission requirements; however, hurricanes have significant spatial and temporal signatures in the visible spectrum, making them ideal to study for an EO mission.

In a natural disaster response scenario, three relevant attributes to system performance are identified: spatial resolution, timeliness, and coverage. These attributes form the basis of the Measures of Effectiveness (MOEs) described below. As discussed in the literature review, spatial resolution is a measure of whether an object of a given size is distinguishable from other nearby objects. It is a means of describing the level of detail in an image, which approximately answers the question, “how useful is this image to an analyst?” In reviewing the literature on the use of imagery in disaster response, the effectiveness of imagery in meeting the responder’s needs was generally described in terms of details detectable at a given resolution in meters.

Timeliness, for this scenario, refers to the amount of time between the natural disaster event (i.e. a hurricane making landfall) and the time a given image is available for an analyst to exploit. As discussed in the literature review, this is generally measured in hours or days, with imagery over 72 hours old being described as “too late” (Department of Homeland Security, 2013). Logically, overall timeliness can be determined from the sum of three sequential factors: time from event to satellite access over the event location for image collect, time from image collect to ground station downlink, and time to process and deliver image to an analyst once downloaded.

Coverage refers to the amount of affected area that can be imaged at a nominal spatial resolution in a given timeline. Typically, the amount of area covered in a single image is limited by field of view; thus, it will take multiple images to investigate the entire affected area, likely over many satellite passes, which could make meeting the timeliness requirement for the affected area difficult. Any architecture that meets the first two MOEs, but only for a small amount of the affected area, is still not meeting the needs of the user.

While not a measure of system effectiveness, cost is a critical aspect of whether or not a system is viable in the domain of budgets and politics. As potential cost savings is a major appeal of a CubeSat architecture, any comparison of CubeSats with other systems would be incomplete without a discussion of cost. This thesis will focus on the systems engineering costs of both architectures, as the MBSE approach for performance analysis described later can facilitate systems engineering cost analysis. Systems engineering cost refers to the systems engineering effort required to realize a system of interest (Valerdi, 2005).

Quantifying Measures of Effectiveness

All three attributes described above have quantitative measures. Spatial resolution can be considered in terms of Ground Resolution or Ground Sample Distance (GSD), with Ground Resolution being a function of a sensor's aperture size, and GSD a function of pixel size and focal length. The literature reviewed discussed resolution almost entirely in terms of GSD; thus, GSD is the measure used in this thesis.

As previously described, GSD is the instantaneous Field of View in one linear dimension for one pixel for a given sensor (Evans, Lange & Schmitz, 2014:184). GSD depends on range and elevation, as well as the design parameters of focal length and detector pitch (sometimes referred to in literature as "pixel size"). The equation used by STK to determine GSD is shown is Equation 1 (Analytical Graphics Incorporated, 2017).

$$GSD = \frac{Detector\ Pitch * Range}{Focal\ Length * \sqrt{\sin(elevation)}} \tag{1}$$

Range and elevation describe the geometry between the satellite and target at a certain point in time; these variables are outputs of the STK simulation for a given image collect.

Overall timeliness is a combination of the length of time (ΔT) of each contributing function of the architecture, from the time a target is affected to the time the imagery is available to an analyst. This relationship is shown in Equation 2.

$$Timeliness = \Delta T_{Target} + \Delta T_{Ground} + \Delta T_{Process} \tag{2}$$

where:

ΔT_{Target} = Time between hurricane landfall and satellite access, with target access windows occurring at night disregarded.
ΔT_{Ground} = Time between image collect and downlink to ground station
$\Delta T_{Process}$ = Time to process an image, from data download until a softcopy image is available to a user.

The two dominant terms in this equation are ΔT_{Target} and ΔT_{Ground}. ΔT_{Target} is a function of event timing and orbital mechanics. It should be noted that event timing, and thus the exact position of the satellite at the time of the event, is a random variable. It is assumed that for hurricanes, the probably distribution is even; a hurricane is just as likely to hit at one given time as it is any other given time. Due to the Earth's rotation, the satellite's orbit is just is as likely to be at one position relative to the target at this given time as it is any other position. To help understand how this variability affects ΔT_{Target}, timeliness and coverage are both calculated for planes at every ascending node, from 0°

to 360°. This allows for the identification of the worst-case scenario for timeliness and coverage, and ensures these MOEs are accounted for given this worst case. Another variable affecting ΔT_{Target} is the timing of daylight at the target, as usable EO imagery cannot be gathered at night; again, this is accounted for in MOE calculations through use of STK to determine whether a given access occurs at day or at night. Access occurring at night are not counted.

ΔT_{Ground} is a function of ground station placement and orbital mechanics. Ground station placement, including number of stations and their locations, is a design parameter; careful consideration of ground station placement in a regional scenario can help minimize ΔT_{Ground}. Again, the time of ground station access for each satellite pass is an output of the STK simulation. In this scenario, $\Delta T_{Process}$ is assumed to have only minor variation between architectures, and is assumed to be negligible compared to ΔT_{Target} and ΔT_{Ground}; this term is not calculated as part of the analysis.

Coverage, the overall amount of area imaged, for a simulated scenario is modeled as shown in Equation 3.

$$\text{Coverage} = A*I*P*\Delta T_{scenario}*N_{sat} \tag{3}$$

where:

A = Area/image
I = Images/satellite pass
P = Satellite passes/unit time
$\Delta T_{scenario}$ = Time of interest between hurricane landfall and scenario end
N_{sat} = Number of satellites

With the assumption of a scanning sensor, the area imaged A is equal to Swath Width times the distance scanned on the ground; a visual representation with scenario-specific details is given in Figure 2. Images per pass is equal to the available target access time in a given pass divided by the amount of imaging time for a single image. Equation 3 assumes no overlap between consecutive images. Determining the number of satellites for each architecture required performing test runs in STK to determine the amount of coverage provided by one satellite, then dividing the coverage requirement by the amount of coverage provided by one satellite, and rounding up to the next integer.

Cost estimation and modeling of traditional satellites is an established field, with models such as SMC's Unmanned Satellite Cost Model (USCM) available to estimate cost based on weight, among other parameters. Cost modeling for CubeSats is less mature; the CubeSat standard is a more recent development, with many missions having been developed by universities. As such, there is little historical data upon which to base a cost model (Selva & Krejci, 2013).

Rather than attempt a new cost model, this thesis will focus on using an existing model for systems engineering costs, COSYSMO, to determine its suitability for comparing systems engineering costs between traditional and CubeSat architectures. Equation 4 shows the highest-level Cost Estimation Relationship (CER) for COSYSMO is defined by Valerdi (2005).

$$PM = A * (Size)^E * \prod_{i=1}^{n} EM_i \tag{4}$$

where:

PM = Person Months
A = Calibration Factor
Size = measure(s) of functional size of a system

E = scale factor(s); default is 1.0
EM_i = effort multiplier for the ith cost driver

This analysis effort is specifically interested in estimating the functional size of each architecture, as this is the parameter MBSE can specifically provide. The general equation for functional size, as defined by Valerdi (2005), is shown in Equation 5.

$$PM_{NS} = \sum_k w_e\Phi_e + w_n\Phi_n + w_d\Phi_d \tag{5}$$

where:

k = requirement/interface/algorithm/operational scenario
w = weight
e = easy
n = nominal
d = difficult
Φ = driver count

This equation is ultimately used to calculated the systems engineering effort required, in terms of person-months.

Design Process

Architecture design was accomplished using OOSEM concepts. OOSEM is a "top-down, scenario-driven process that uses SysML to support the analysis, specification, design, and verification of systems" (Friedenthal, Moore, & Steiner, 2015). This thesis specifically used the system specification and design process from OOSEM. As described by Friendenthal, Moore and Steiner (2015), the design process consists of five steps:

1. Set up model
2. Analyze stakeholder needs

3. Analyze system requirements
4. Define logical architecture
5. Synthesize candidate physical architectures

Figure 1 shows a mapping of steps taken for this thesis to the OOSEM system specification and design process, starting with step 2. The bottom arrow from the "Architecture Performance/Cost Analysis" block back to the "Analyze System Requirements Block" and "Synthesize Candidate Architecture" block illustrates the iterative nature of this process.

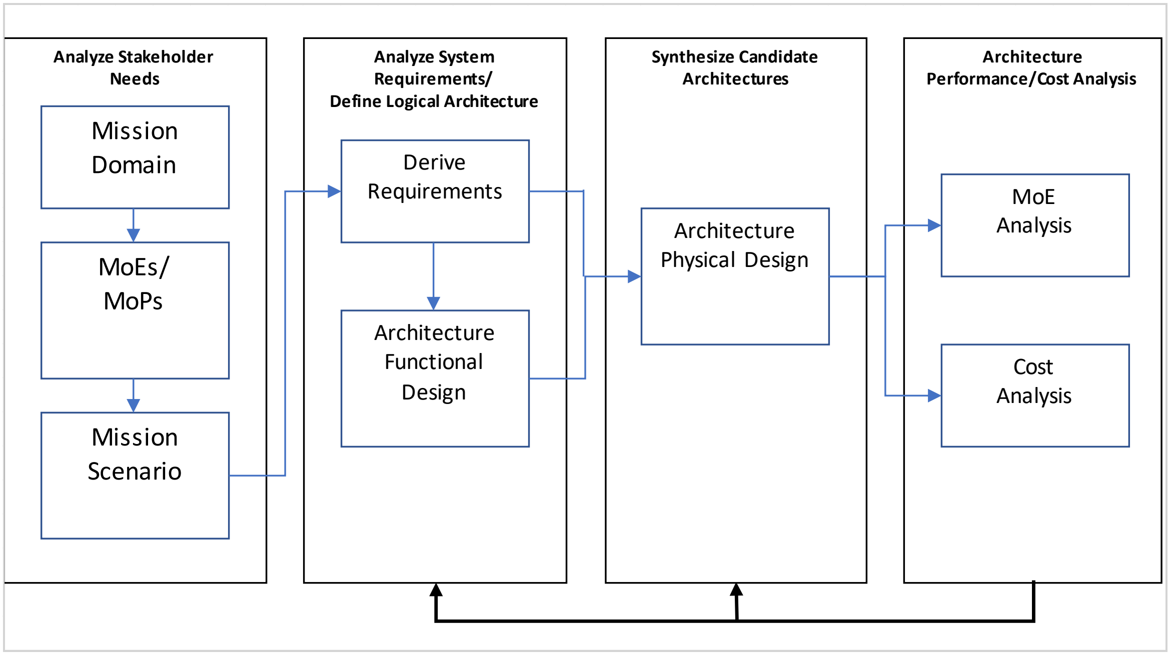

Analyze Stakeholder Needs
Mission Domain
MoEs/ MoPs
Mission Scenario
Analyze System Requirements/ Define Logical Architecture
Derive Requirements
Architecture Functional Design
Synthesize Candidate Architectures
Architecture Physical Design
Architecture Performance/Cost Analysis
MoE Analysis
Cost Analysis

Figure 1. Thesis Steps Mapped to OOSEM Process

Scenario for Architecture Performance Analysis

To develop realistic mission requirements based on stakeholder needs, a mission scenario is necessary. As mentioned above, the fact that hurricanes are spatially large, temporally long (both the phenomenon itself and its impact), and spatially and temporally dynamic assists in gaining an understanding of how well an architecture addresses a mission need.

The scenario for this thesis is loosely based on 2017's Hurricane Maria. On 20 September 2017, Hurricane Maria made landfall on Puerto Rico as a strong Category 4, with winds up to 155 mph. Hurricane Maria affected the entire island, causing extensive damage to buildings and infrastructure, and creating serious flooding concerns (Schmidt, Achenbach, & Somashekhar, 2017).

In the thesis scenario, it is the objective of disaster response personnel to use satellite imagery, ideally with before-and-after change detection, to identify damage to structures and infrastructure, and to identify areas of flooding across the entire island within 72 hours after landfall. Some adjustments to and assumptions for this scenario are necessary. The time period for this scenario, placed near the peak of hurricane season, was arbitrarily chosen as 0000 UTC 11 August to 0000 UTC 18 August, 2017, with the first 72 hours being of particular interest. These dates are hard-coded into the Python scripts used to analyze MOEs; exact choice of dates is assumed to not significantly affect the results. The area of interest is the entire island of Puerto Rico, which is 177.8 km by 64.8 km, or roughly 11522 square km. Puerto Rico was modeled in STK as a point target centered at 18.22° N, 66.57° E (Google, n.d.). With a relatively low latitude compared to other US locations, Puerto Rico also becomes a more stressing case for timeliness and coverage, as access for spacecraft in sun-synchronous orbits typical for remote sensing is less frequent.

In order to determine satellite coverage per pass, it was assumed that satellites moved North to South or South to North, so that the length of the area of interest was the width of the island at its widest point, 64.8 km. Figure 2 illustrates the coverage area per pass:

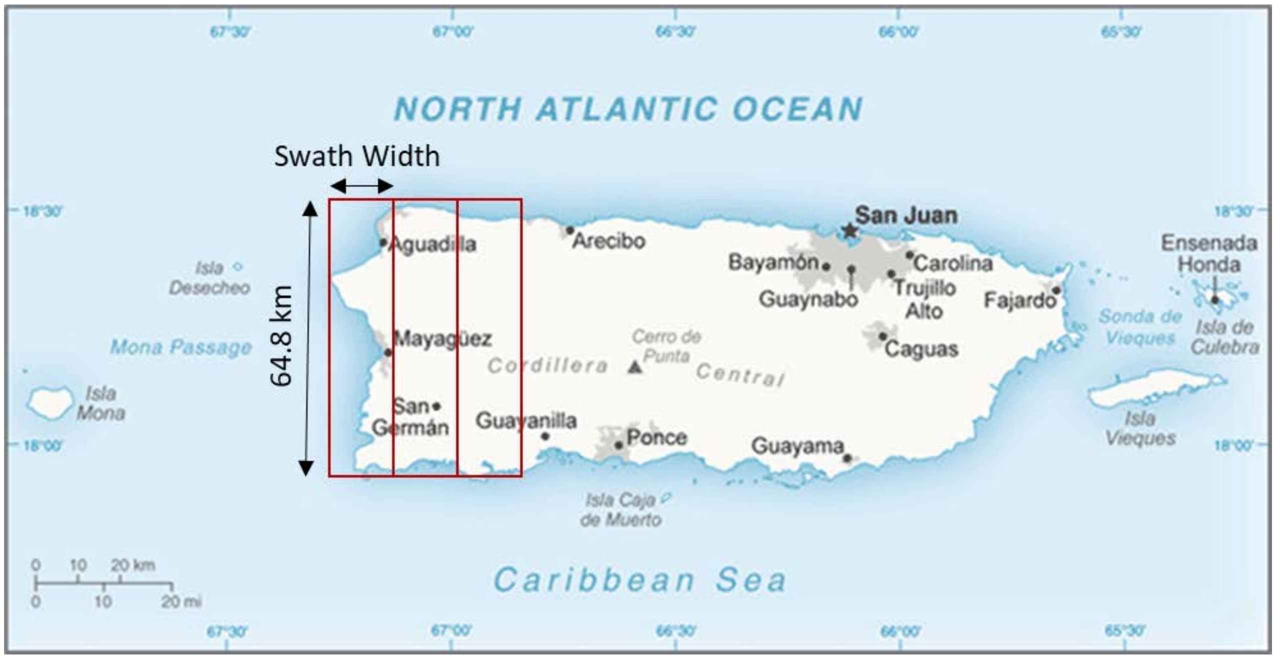


Figure 2. Determination of Coverage Per Pass Over Puerto Rico. Modified from (Central Intelligence Agency, 2017)

Locations outside of Puerto Rico impacted by Maria were not considered for simplicity. It is assumed that flooding in this scenario can be detected using imagery better than 10 m GSD, and damage to individual buildings can be detected at imagery better than 1 m GSD. It is assumed that change detection greatly aids in the accomplishment of identifying flooding and damage, but that this could also be accomplished without.

Developing Architectures from Requirements

With MOEs and a mission scenario defined, architectures to accomplish this mission were developed and modeled. This process begins with mission requirements, which is based on the mission scenario above and will be common to both architectures.

Mission requirement values are outlined in Table 2.

Table 2. Values for Attributes Driving Mission Requirements

Attribute	Threshold	Objective	Units
Spatial Resolution	10	1	Meters
Timeliness	72	24	Hours
Coverage	11522	Same	Square kilometers
Access	37°	All	Degrees latitude
Change Detection	Must be capable	N/A	N/A

These requirement values were derived from values commonly found during the literature review, as discussed above. Verbiage for all requirements is captured in SysML requirement diagrams and accompanying tables in the Cameo Systems Modeler tool.

After mission requirements were determined, architecture-specific design and modeling was accomplished. The design and modeling processes are iterative in nature and occurred in parallel. These processes were common to both traditional and CubeSat solutions; however, for this thesis, the traditional architecture was designed and modeled first. The traditional architecture model then became a conceptual template for the CubeSat architecture design, with CubeSat-specific modifications to the requirements and design solution made as necessary. Each architecture consists of five chief aspects: use cases, requirements, physical elements, interfaces, and algorithms. The reasoning for this is discussed further in the “Modeling Process” section.

To begin the architecture design process, use cases were written to describe usage scenarios. Most use cases focused on the system actions required to accomplish mission requirements. Use cases for system support and off-nominal situations were written as well. Use cases were written with specific attention paid to the functions the system would need to perform; these functions, once identified, became the basis for segment-level functional requirements.

With use cases and high-level functions identified, the next step was to develop, at an abstract level, some degree of notional physical implementation. Beneath the system level, each architecture was broken down into segments, then broken down further into components. Functional requirements were parsed out to these segments, and then further decomposed for individual components of each segment to satisfy.

Once components were established, necessary interfaces between components were identified. At this level of abstraction, data interfaces were the most relevant; electrical power interfaces between satellite components were also considered.

For the purposes of COSYSMO cost modeling, system-specific algorithms are defined as “new defined or significantly altered functions that require unique mathematical algorithms to be derived to achieve system performance requirements” (Valerdi, 2005). Applying that definition to the architectures of interest, an algorithm is identified anywhere a function is performed, at the system or component level, that transforms one or more data inputs into data outputs. It is assumed that functions identified as algorithms would be performed via software implementation.

Architecture Design Details

Starting from the mission requirements and a general knowledge of imaging systems, thought was given as to how the system would be used operationally, and which external actors would interact with the system during operations. These thoughts are captured as text in the form of use cases. The use cases pertinent to the traditional architecture are shown in Figure 3:

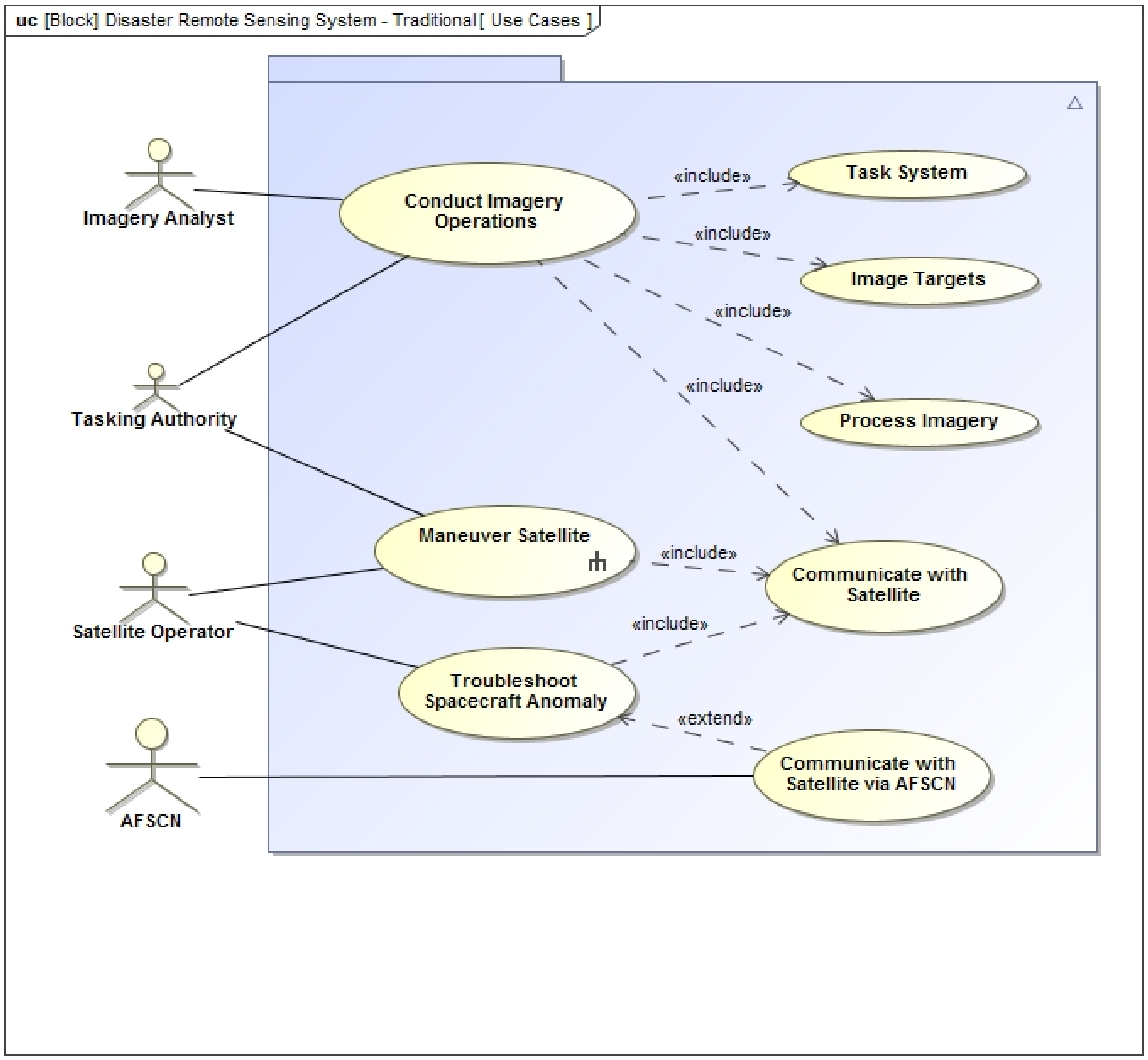

uc [Block] Disaster Remote Sensing System - Traditional [Use Cases]
Imagery Analyst
Tasking Authority
Satellite Operator
AFSCN
Conduct Imagery Operations
Task System
Image Targets
Process Imagery
Maneuver Satellite
Communicate with Satellite
Troubleshoot Spacecraft Anomaly
Communicate with Satellite via AFSCN
«include»
«extend»

Figure 3. Use Case Diagram for the Traditional Architecture

The primary mission of this system is to *conduct imagery operations*. This use case includes three main functions, captured as "include" use cases: *sensor tasking, imagery collection*, *imagery processing*, and *imagery delivery*. These functions are derived from the "Tasking," "Collection," and "Processing" steps of the Tasking, Collection, Processing, Evaluation, and Dissemination (TCPED) process.

Two “support” use cases, used for system support but not directly used for mission accomplishment, are *maneuver satellite* and *troubleshoot spacecraft anomaly*. The use cases stem from the system-level design life requirement, along with derived requirements for satellite stationkeeping and anomaly recovery. All three main use cases require the include use case *communicate with satellite*; a back-up communication capability is described in the *communicate with satellite via AFSCN*[2] use case. Full text for the traditional architecture use cases is provided in Appendix A.

The CubeSat architecture makes use of the same mission-related use cases as the Traditional architecture; however, there are differences in the support use cases, as shown in Figure 4.

[2] AFSCN: Air Force Satellite Control Network

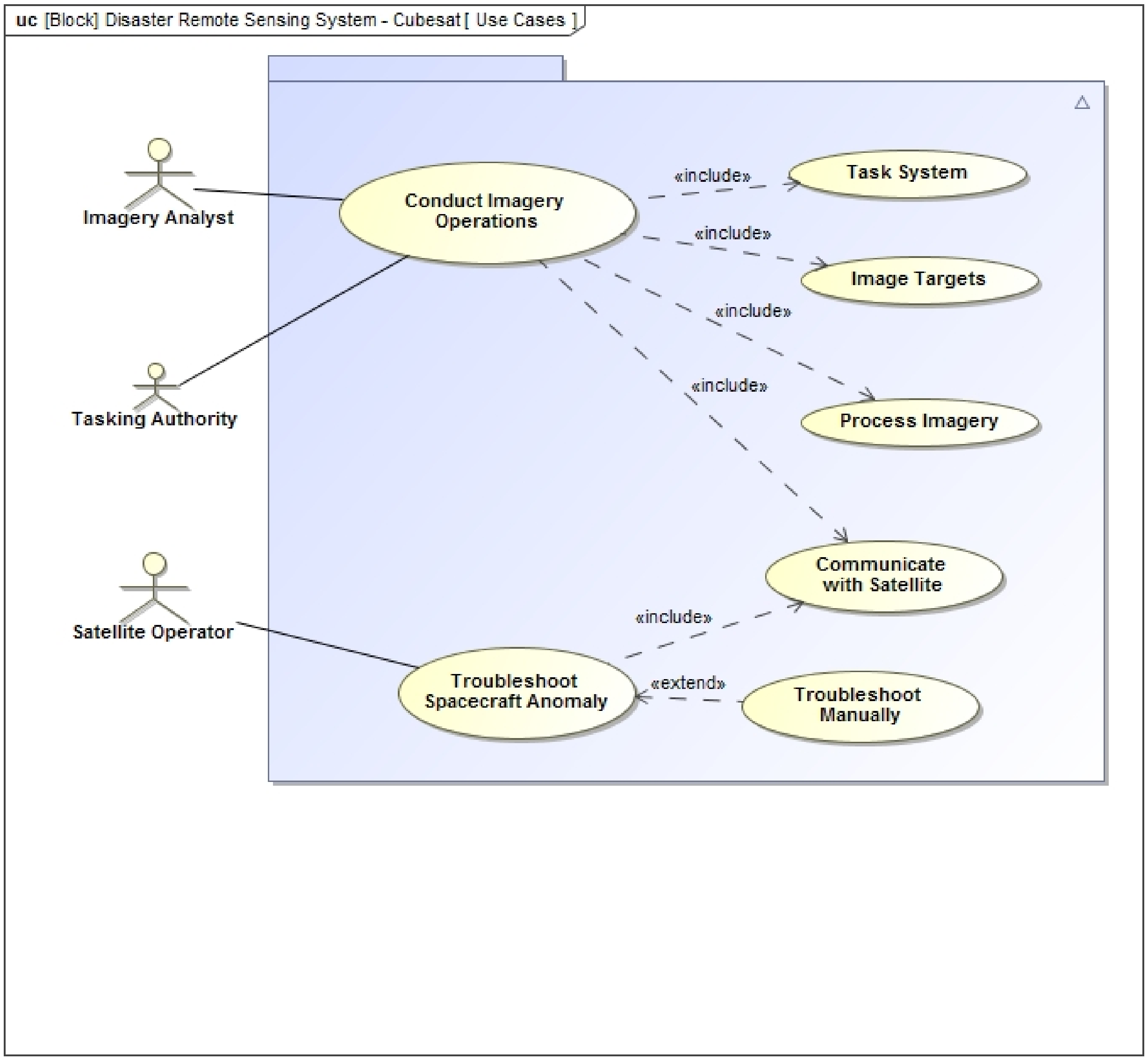


Figure 4. Use Case Diagram for the CubeSat Architecture

One support use case has been removed entirely: for the CubeSat architecture, the design life requirement is relaxed from seven years to one, negating the need for *maneuver satellite*. Additionally, there is no need for the *communicate with satellite via AFSCN* use case, as no AFSCN backup communication capability is envisioned. The CubeSat architecture does have a new extend use case, *troubleshoot manually*, as it is envisioned that there would be separate steps that both the spacecraft and the spacecraft operator could go through to resolve the off-nominal condition. Full text for the CubeSat architecture use cases is provided in Appendix B.

These use cases form the basis for deriving functional requirements at the segment and component level, followed by performance requirements and design constraints. There is significant overlap between the requirements for the two architectures; only key differences are highlighted in this discussion. The key driving difference is a design constraint: the traditional architecture satellite is required to fit inside an Evolved Expendable Launch Vehicle (EELV) payload fairing, while each CubeSat is required to conform to a 6U standard size. Other requirement changes stem from the use cases mentioned above; specifically, the CubeSat architecture has no requirements for a propulsion system or connectivity to the AFSCN. It should be noted that performance requirements that are derived from mission requirements, such as resolution and coverage, are not relaxed for the CubeSat architecture.

Moving from requirements to physical descriptions of the design, each architecture consist of three segments: a Command and Control (C2) segment, a satellite segment, and an imaging processing segment. The C2 segment performs tasking and commanding functions, and provides the interface between a Tasking Authority actor who requests imagery and the satellite collecting imagery. Additionally, the C2 segment provides support functions such as commanding for orbital maneuvering and telemetry processing and display. The physical structure of the C2 segment, shown in Figure 5, does not vary between architectures. It should be noted that the traditional architecture C2 segment does have an additional interface, from the C2 processor to the external AFSCN.

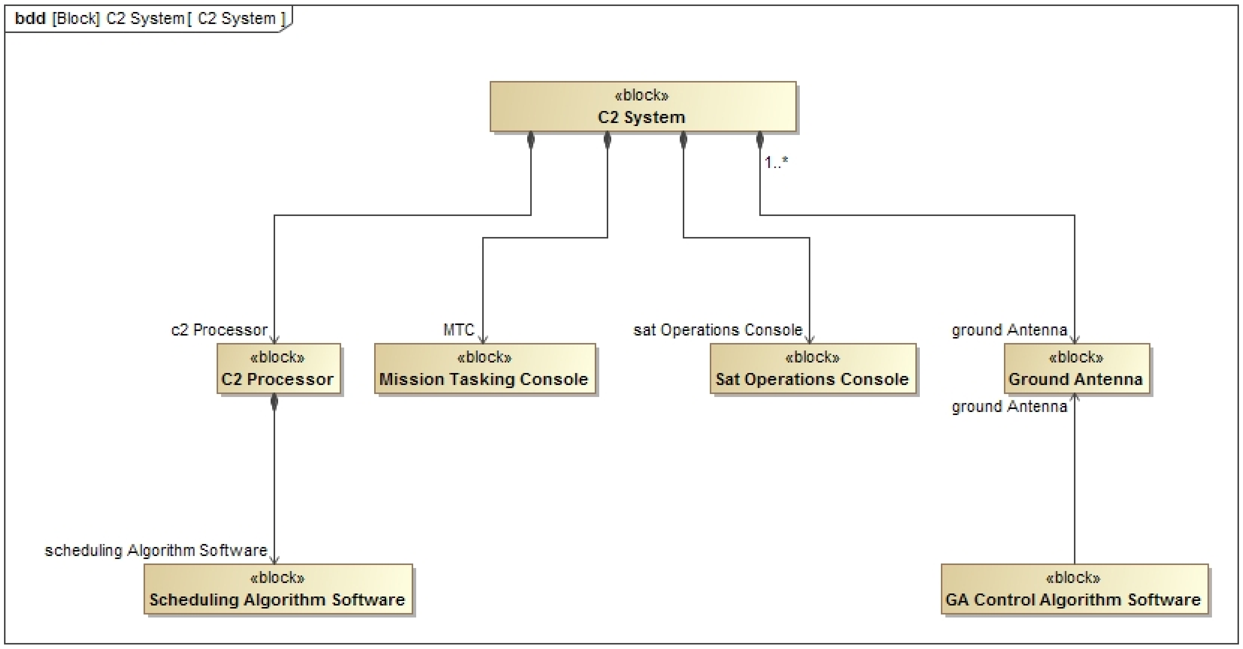


Figure 5. Command and Control Segment Block Diagram

A key consideration for the C2 segment was the number and location of ground antennas. This design decision was driven by two constraint requirements: 1) All image data from a satellite pass had to be downlinked prior to that satellite's next image window, and 2) The ground antenna(s) could not be located within 250 miles of coastline prone to hurricanes. To meet the first requirement, ground stations were located such that every satellite could downlink its images within one pass. The easiest way to accomplish this was to locate one ground station at a very high latitude towards either the North or South pole, so that all satellites had access on every pass. An antenna placed at an existing NASA ground site in Svalbard, Norway was chosen for both architectures.

Three relevant algorithms were identified in the traditional architecture C2 segment: a scheduling algorithm, a ground antenna control algorithm, and a maneuvering algorithm. The scheduling algorithm takes image tasking and turns it into executable commands to be sent to the satellite. The ground antenna control algorithm provides steering control for the ground antenna to maintain contact with the spacecraft as it passes overhead. The maneuvering algorithm calculates the necessary orbit adjustments for stationkeeping and turns it into executable commands to be sent to the satellite. Each of these algorithms would likely consist of many sub-algorithms; however, for the purposes of this thesis, it is only necessary to specify which high-level algorithms are necessary to fulfill functional requirements. The CubeSat C2 segment has one fewer algorithm, as the software functionality for maneuvering is not needed.

The traditional satellite segment design is based on an arrangement of components, usually called subsystems, commonly found on existing satellite designs. Modifying the subsystem arrangement given by Wertz and Larson (2010), these subsystems are the payload, Attitude Determination and Control (ADCS), communications, Command and Data Handling (CD & H), power, structures, and propulsion[3]. These subsystems were only developed to the level of detail needed to distinguish a traditional design from a CubeSat design.

As mentioned earlier, a key differentiator between architectures is the size constraint; this subsequently limits payload volume. In addition to physical size, both the imagery resolution requirement and the coverage requirement drive sensor design parameters. GSD is a function of pixel size and focal length (Krueger, 2009); coverage is a function of swath width, which geometrically is a function of the sensor Field of View (FOV). To avoid a complicated sensor design problem within this thesis, the design

[3] Wertz and Larson list three other subsystems: thermal, guidance, and computer systems. Thermal is excluded in both architectures, as it is a support subsystem not expected to be a major differentiator between architectures. ADCS is assumed to perform any guidance functions, and computers/software are split between CD&H, ADCS, and ground systems.

parameters of a satellite camera known to satisfy the spatial resolution requirement were chosen. The WV110 camera, flown aboard the Worldview-2 satellite, has a GSD of 0.46 m at nadir at an altitude of 770 km (European Space Agency, 2017). Key parameters of the WV110 are shown in Table 3.

Table 3. Design Parameters of the WV110 Camera (European Space Agency, 2017)

Parameter	Value
Nadir GSD at 770 km altitude	0.46 meters
Aperture Diameter	1.1 meters
Focal Length	13.3 meters
Field of View	>1.28°
Panchromatic Spectral Range	450 to 800 nanometers
CCD Detector Pixel Size	8 micrometers
Data Quantization	11 bits
Data Compression	2.75 bits/pixel

For the CubeSat architecture, the requirement to fit inside a 6U standard structure placed inherent limits on sensor dimensions such as focal length. Again, to avoid a complicated design problem, the CubeSat sensor parameters are taken from a pre-existing design. Table 4 lists the design parameters for Planet Lab's Planet Scope 2 EO sensor, which is flown aboard Planet Lab's Flock series of 2.5U CubeSats.

Table 4. Design Parameters of the PS2 Camera (Planet, 2015; Boshuizen et. al, 2014)

Parameter	Value
Nadir GSD at 475 km altitude	3.73 meters
Aperture Diameter	approx. 0.1 meters
Focal Length	1.14 meters
Field of View	>1.94°
Panchromatic Spectral Range	420 to 700 nanometers, 3 bands
CCD Detector Pixel Size	8.95[4] micrometers

[4] Value not explicitly stated in literature, but calculated using given focal length, altitude, and GSD at nadir.

For this architecture, after the payload, communication is considered the next design driver, as it is the connecting piece between the payload and the ground. To minimize complexity in this design, the imaging and communications functions do not occur simultaneously. The satellite takes images of a target, and then stores that data for download at the next communications, or “comm,” window with a ground antenna. The communications subsystem for this architecture consists of two antennas: a directional antenna for primary communications with the system’s ground antenna, and a backup omni antenna for communication with the AFSCN if the primary communications link is lost.

For the CubeSat architecture, the communications subsystem performs the same role, with one major design change: there is no omni antenna. Given the much more limited space and the shorter required design life of the CubeSat architecture, a backup communications capability was not added. It is recognized that the implementation of the main communication antenna, both on the spacecraft and ground, would vary between architectures in terms of antenna size, power, and required data rate; however, for simplicity, this subsystem was not designed to that level of detail. It is assumed that a plausible antenna design solution meeting requirements exists for each architecture; with this assumption, this communications design has no impact on the MOEs.

For the remaining subsystems, design focused on the primary functions each subsystem accomplished, and the data flow necessary to accomplish those functions. The focus on data allowed the determination of both necessary interfaces and necessary software algorithms. For example, the ADCS subsystem for both architectures requires an interface with the CD&H subsystem to receive a desired attitude vector; the ADCS then uses an algorithm to determine current orientation and the correct series of attitude adjustments to reach the desired attitude. Given that satellites in both architectures both require nearly the same support functions from these subsystems, the designs at this level do not vary, with one exception. As previously mentioned, the CubeSats in this design have no propulsion subsystem, as maneuvering/stationkeeping is not required.

In addition to the previously mentioned ADCS algorithm, both architectures have a telemetry monitoring algorithm as part of the CD&H subsystem. This algorithm automatically detects damaging spacecraft conditions and puts the spacecraft in a protective safe mode when necessary. In order to keep a consistent, simple design, many software functions are provided by ground segments instead of onboard; for example, image processing is done by the image processing segment, with spacecraft functions limited to collecting and transmitting raw “mission data”.

With designs for individual satellites established, the next step was to determine necessary orbital parameters. For simplicity, all orbits are approximated as circular with an eccentricity of 0°. Both architectures have a change detection requirement; meeting this requirement necessitates a sun-synchronous orbit, with near 98° inclination and an altitude between 475 and 800 km[5]. This orbit also ensures both architectures meet the access requirement of 37° latitude or higher. For both architectures, there is a trade-off between better GSD at low altitude and better coverage at higher altitudes. For the CubeSat, the payload size constraint and subsequent limitations on sensor performance meant keeping the satellite as low as possible; thus, altitude was set to 475 km. For the

[5] Sellers (2005:164) defines sun-synch as approximately 150 to 900 km altitude. The range for this thesis is narrower to recognize that altitudes lower than 475 km encounter more orbital drag, and higher altitudes negatively impact GSD.

traditional architecture, there was less concern about meeting the GSD threshold requirement in this altitude range; this opened up trade space to either maximize coverage by setting the altitude to 800 km, or maximize GSD by setting the altitude to 475 km. Preliminary STK simulations at both altitudes showed that the coverage requirement was easily met at 475 km, thus this value was chosen[6]. At these altitudes, the inclination for sun-synchronous is approximately 97.9°. Details of choosing ascending node and mean anomaly are discussed in the modeling process section. After running test simulations in STK to determine coverage per satellite over 72 hours, it was determined that an architecture of one traditional satellite or three CubeSat satellites could meet the coverage requirement. Both architectures are limited to a single orbital plane; the traditional architecture by default, and the CubeSat architecture in recognition that multiple planes would require either a propulsion system or multiple launch vehicles.

The third segment is the Imaging Processing Segment (IPS), consisting of an imaging processor and a storage database. This segment provides the capability to ingest mission data, turn mission data into an image, turn two images into a change detection product, and store image and change detection products for retrieval. The IPS also provides the interface between the system of interest and whatever means an imagery analyst uses to exploit the imagery, though it does not provide the image viewing capability itself. As with the C2 segment, the structure of the IPS, shown in Figure 6, does not vary between architectures.

[6] Results are further discussed in Chapter 4.

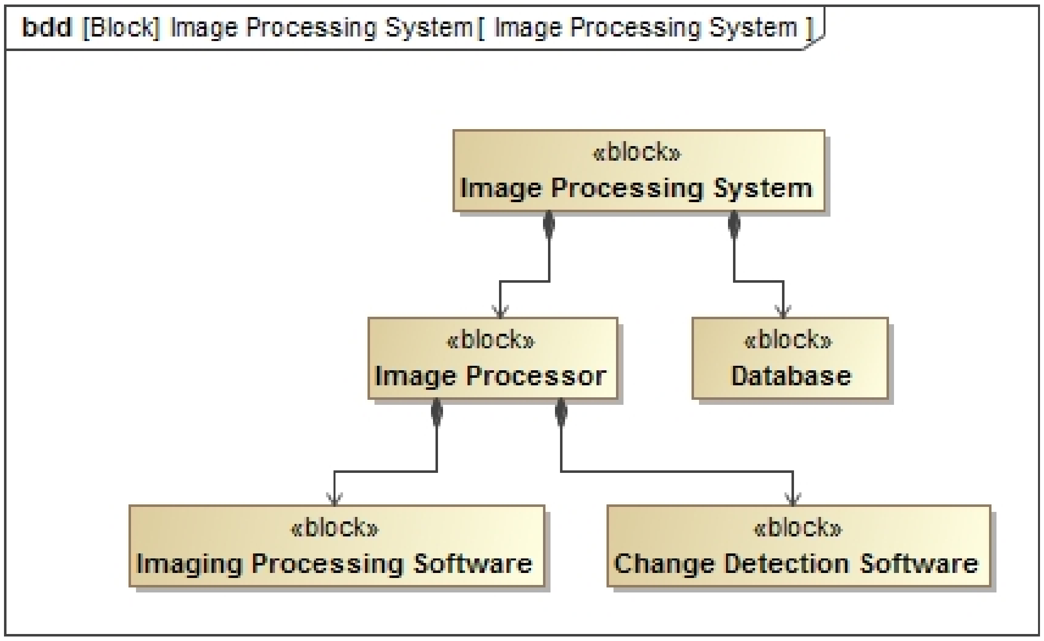

bdd [Block] Image Processing System [Image Processing System]
«block»
Image Processing System
«block»
Image Processor
«block»
Database
«block»
Imaging Processing Software
«block»
Change Detection Software

Figure 6. Image Processing Segment Block Diagram

In terms of software, the IPS consists of two main algorithms: an image processing algorithm and a change detection algorithm. The image processing algorithm turns mission data downlinked from the spacecraft into an interpretable image. The change detection algorithm takes two images, notionally from before and after the disaster event, and identifies portions of the image that have changed.

This section has described the design details of the architectures to be modeled and analyzed, with key differences between the architectures highlighted. These differences are summarized in Table 5.

Table 5. Design Differences Between Traditional and CubeSat Architectures

Design Aspect	Traditional	CubeSat
Physical size	Fits within EELV payload fairing	6U or smaller
Design life	7 years	1 year
Camera	See Table 3	See Table 4
Comm subsystem	Directional and omni antennas	Directional antenna
Propulsion subsystem?	Yes	No
Number of satellites in constellation	1	3

Modeling Process

The purpose of modeling these architectures is twofold: first, to systematically derive and define parameters to input into STK for performance modeling; second, to provide inputs to COSYSMO for cost modeling. The purpose for modeling defines the views to be developed.

MBSE projects involve three upfront decisions: choice of method, choice of language, and choice of tool (Delligatti, 2014:4). As mentioned previously, the Object-oriented Systems Engineering Method (OOSEM) was chosen as the method. The OOSEM system specification and design process provided a structured and logical way to derive architectures from stakeholder needs. The language chosen for modeling in this thesis is SysML. SysML is commonly used for MBSE (Delligatti, 2014:5); its common usage and the availability of resources related to it made it a logical choice. The primary tool for architecture modeling was Cameo Systems Modeler, version 18.5. Vitech's CORE systems modeling software was considered and ruled out due to limitations in the provided educational license. Additionally, Cameo easily allows for architectures to be captured as XML files; this was helpful for parsing the architectures for COSYSMO input parameters.

Modeling views to be developed were modified from Edwards (2016). Edwards' method makes use of block, package, requirements, internal block, parametric, and use case diagrams to parse SysML for COSYSMO input. For this thesis, the views developed and their purpose is described in Table 6:

Table 6. Architectures Views and Purposes

View	View Purpose/Information conveyed
Requirements Table and Diagrams	Captures system, functional and physical requirements
Block Definition Diagram	Logical decomposition; identifies relevant hardware and software components
Internal Block Diagram	Models interfaces between components
Use Case Diagram	Captures operational scenarios from use cases

Before modeling in Cameo, use cases were written out as text. Once sufficiently understood, the use cases for each architecture were captured in a use case diagram. The requirements table feature in Cameo became the primary means of capturing requirements. Requirements were written at the system level, then further derived to the segment and component/subsystem level. Requirement diagrams were generated to help visualize the relations between requirements, but were not strictly necessary for analysis. Interfaces were modeled between the system and external systems, between system segments, and between system components. In Cameo, these interfaces were modeled as port elements belonging to the components comprising the interface.

Major algorithms in each architecture are assumed to be implemented via software. Individual software components representing each algorithm are modeled as blocks, as shown in the Block Definition Diagrams featured in Figures 5 and 6. When modeling needed software components, it was helpful to identify and model required data items as separate blocks. Data items themselves are not input for any of the analysis in this thesis, but having them modeled as distinct blocks provided clarity and made development of component block diagrams easier. A one-to-one mapping between algorithms and software components is assumed; a given block of software does not perform more than one algorithm function, or vis-versa.

Simulation and Analysis Process

Once the traditional and CubeSat architectures were sufficiently designed and modeled, parameters from each model provided input for performance and cost analysis. Performance analysis was accomplished through a combination of STK simulation and Python scripts. Simulation set-up was accomplished with a Python script; this script generated satellite, target, and ground station[7] instances, then passed them to STK through the software's Connect module. The pertinent design parameters are shown in Table 7.

[7] In STK, Ground Stations are modeled as "Facility" objects.

Table 7. Design Parameters for STK Simulation Input[8]

Parameter	Traditional Architecture	CubeSat
Scenario Start Date/Time	11 Aug 2017 00:00:00.000	11 Aug 2017 00:00:00.000
Scenario End Date/Time	18 Aug 2017 00:00:00.000	18 Aug 2017 00:00:00.000
Target lat/long	18.22°N, 66.57°W	18.22°N, 66.57°W
Number of Satellites	1	3
Altitude	475 km	475 km
Inclination	97.9°	97.9°
Eccentricity	0	0
RAAN	0° - 359°	0° - 359°
Mean Anomaly	0°	0°, 120°, 240°
Ground Station lat/long	78.23°N, 15.38°E	78.23°N, 15.38°E

The STK simulation generated two types of output products: access reports and Azimuth/Elevation/Range (AER) reports. These reports were generated for each combination of satellite and ground target or facility. Access reports provided start and stop times for satellite access to a target or facility; target accesses correspond to "image windows" and facility accesses correspond to "comm windows". AER reports provided azimuth/elevation/range values for every minute of access. This provided a convenient measure of images taken per image window; it could be assumed that the satellite took an image and then slewed to take another image on a 60-second timeline. Additionally, each simulation generated a lighting report, denoting sunrise and sunset times for the target.

One limitation of STK is scheduling; in a simulation, if two targets are within a satellite's field of regard at the same time, the satellite will capture the first target that comes into view, and continue imaging this target until it is out of view. Only after the first target is out of view will the satellite switch to a second target. To work around this

[8] STK also requires an Argument of Perigee; however, for a circular orbit, the value of this parameter is arbitrary

limitation, the entire area of interest is modeled in STK as a single point target on or near the geometric center of Puerto Rico, shown in Figure 7.

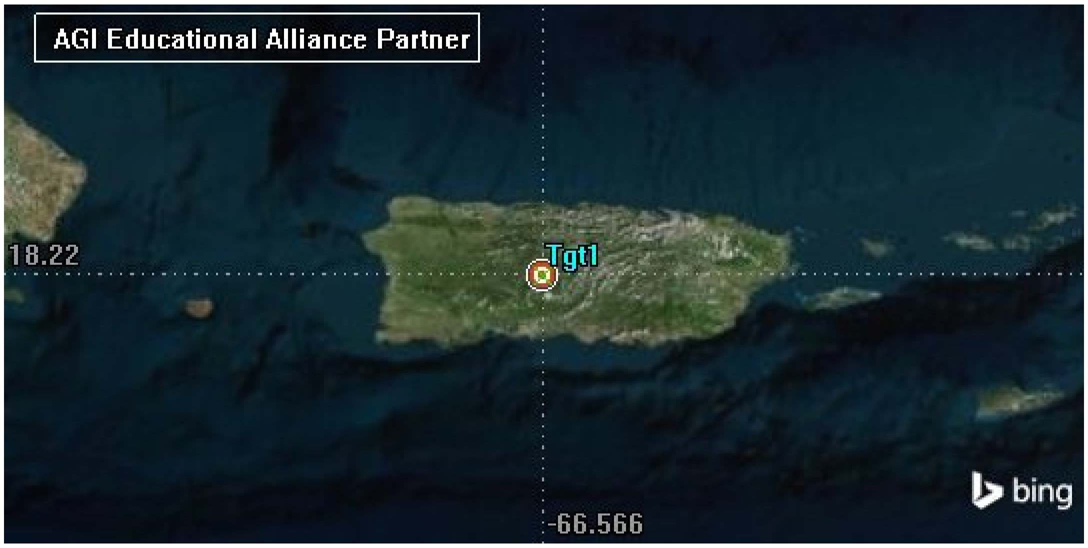


Figure 7. Point Target at Geographic Center of Puerto Rico

Although the access and AER reports are based on the location of this point target, it is assumed that on each pass the satellite would actually image one or more "strips" of area, as shown previously in Figure 2. The differences in range and elevation for a given collect caused by this assumption are presumed to be negligible for an area this small.

For post-simulation analysis, two Python scripts were written; one to calculate the spatial resolution MOE, and one to calculate the timelines and coverage MOEs. Additional design parameters needed for post-STK analysis are listed in Table 8.

Table 8. Design Parameters for Post-STK Simulation Input

Parameter	Traditional Architecture	CubeSat
Focal Length	13.3 m	1.14 m
Pixel Size	8 μm	8.95 μm
Swath Width	17.87 km	16.1 km

Figure 8 describes the steps for calculating the resolution MOE for a given architecture. Five parameters were calculated for resolution: best (or minimum) GSD, worst (or maximum) GSD, average GSD, percent of collects meeting the threshold requirement, and percent of collects meeting the objective requirement. Calculations were performed for one satellite in each architecture, on all data points from that satellite, regardless of daylight conditions. Calculating resolution from identical satellites and parsing out daylight-only collects would have had minimal to no effect on the overall aggregate results.

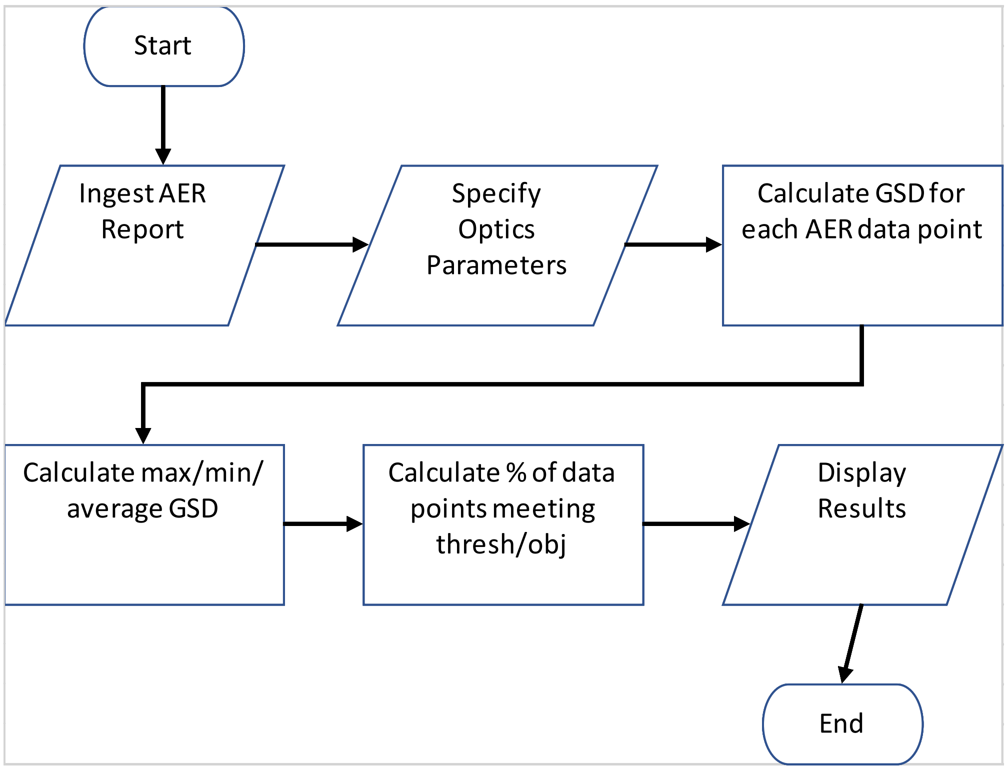

Start
Ingest AER Report
Specify Optics Parameters
Calculate GSD for each AER data point
Calculate max/min/ average GSD
Calculate % of data points meeting thresh/obj
Display Results
End

Figure 8. Calculation of Resolution MOE

Figure 9 below shows the general flow of the timeliness/coverage MOE analysis script. In this script, each individual image window, and eventually each individual image, is treated as an instance of an "image window" or "image collect" object. Unlike the resolution MOE, the timeliness and coverage MOEs are calculated for each satellite in each architecture. Additionally, this script is run for each simulated ascending node, to identify timeliness and coverage for the best-, average, and worst-case target/orbit combinations.

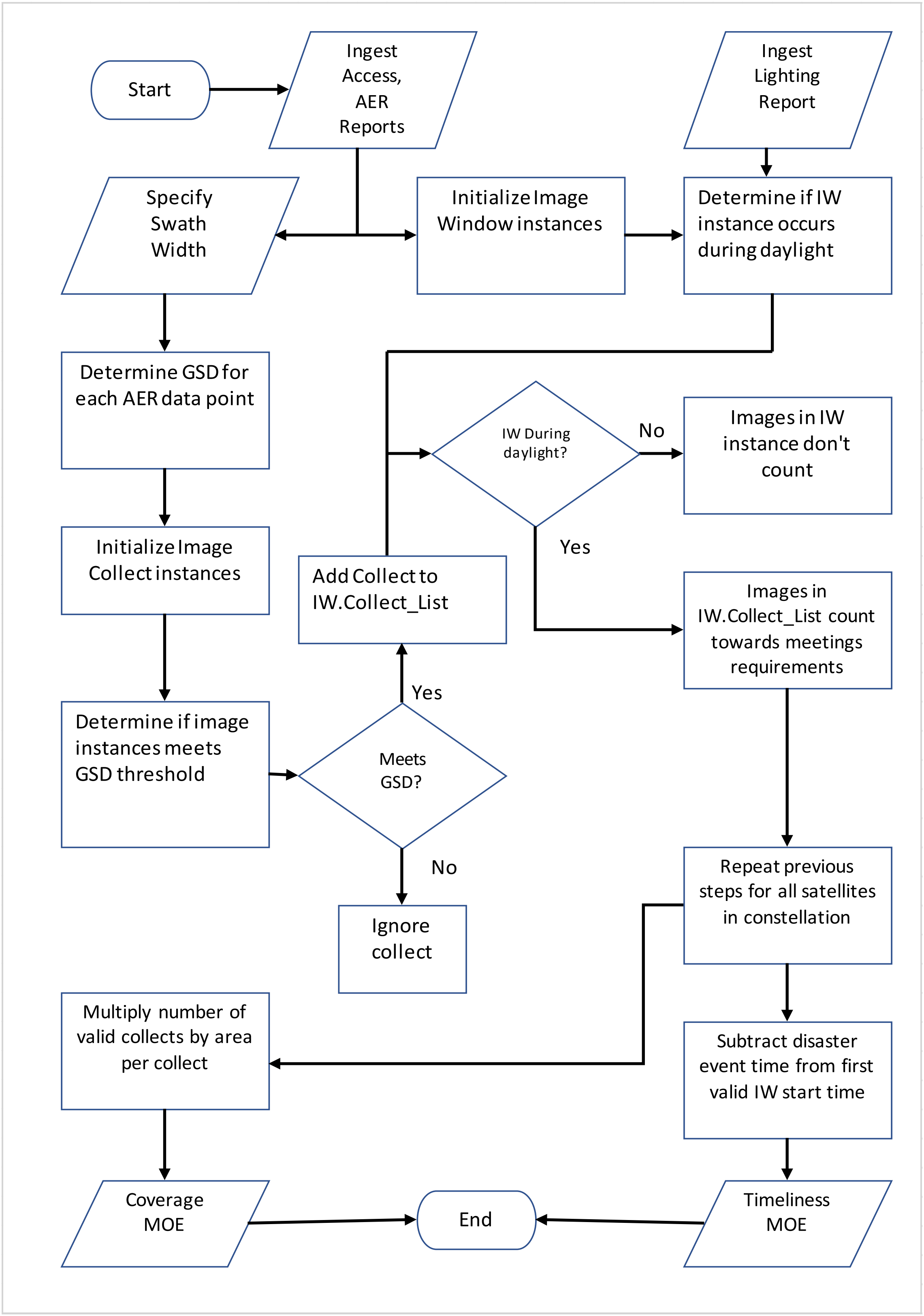

Start
Ingest Access, AER Reports
Ingest Lighting Report
Specify Swath Width
Initialize Image Window instances
Determine if IW instance occurs during daylight
Determine GSD for each AER data point
IW During daylight?
No
Images in IW instance don't count
Yes
Initialize Image Collect instances
Add Collect to IW.Collect_List
Images in IW.Collect_List count towards meetings requirements
Yes
Determine if image instances meets GSD threshold
Meets GSD?
No
Repeat previous steps for all satellites in constellation
Ignore collect
Multiply number of valid collects by area per collect
Subtract disaster event time from first valid IW start time
Coverage MOE
End
Timeliness MOE

Figure 9. Calculation of Timeliness and Coverage MOEs

To obtain and analyze cost estimates from COSYSMO, relevant data was parsed from the Cameo model of each architecture. As described earlier, this data includes counts of the requirements, interfaces, algorithms, and use cases in each architecture model. Each of these items is given an assessment of easy, nominal, or difficult; a summary of definitions of these terms as provided by Valerdi (2005) is given in Table 9. The size driver counts from both architectures and their associated difficulty assessments were input into the COSYSMO function of the Naval Postgraduate School's (NPS) System Cost Model Suite, which calculated each architecture's functional size using equation (5)[9].

[9] This step was performed by faculty at the Naval Postgraduate School, with results e-mailed to the student and to AFIT faculty.

Table 9. Size Driver Difficulty Rating Definitions (Valerdi, 2005)

Requirements		
Easy	**Nominal**	**Difficult**
Simple to implement, Traceable to source, Little requirements overlap	Familiar, Can be traced to source with some effort, Some overlap	Complex to implement or engineer, Hard to trace to source, High degree of requirements overlap
Interfaces		
Easy	**Nominal**	**Difficult**
Simple message, Uncoupled, Well behaved	Moderate complexity, Loosely coupled, Predictable behavior	Complex protocol(s), Highly coupled, Poorly behaved
Algorithms		
Easy	**Nominal**	**Difficult**
Algebraic, Straightforward structure, Simple data, Timing not an issue, Adaptation of library-based solution	Straightforward calculus, Nested structure with decision logic, Timing a Constraint, Some modeling involved	Complex constrained optimization/pattern recognition, Recursive in structure with distributed control, Noisy/ill-conditioned data, Dynamic, with timing and uncertainty issues, Simulation and modeling involved
Operational Scenarios		
Easy	**Nominal**	**Difficult**
Well defined, Loosely coupled, Timelines not an issue, Few and simple off-nominal threads	Loosely defined, Moderately coupled, Timelines a constraint, Moderate number or complexity of off-nominal threads	Ill-defined, Tightly coupled, Tight timelines, Many or very complex off-nominal threads

Summary

This chapter described the methodology used in this thesis, outlining how Model-Based Systems Engineering is used to develop SysML models of traditional and CubeSat imagery architectures. These models provide inputs into the STK performance simulation and COSYSMO cost estimation tools in order to compare the two architectures in terms of performance and cost.

IV. Analysis and Results

Chapter Overview

This section discusses the results of the simulation and analysis described in the previous chapter. This section seeks to address the investigative questions described in Chapter Two, which were:

1. Given a set of mission objectives and requirements, how well does a CubeSat and a traditional remote sensing architecture meet these requirements?
2. Are systems engineering cost models such as COSYSMO a useful means of predicting systems engineering costs for traditional and CubeSat architectures? Does it provide a valid means of comparison?
3. What are the implications of using MBSE to answer questions 1 and 2?

Question 1: Results of Performance Analysis

Spatial Resolution

The highest, lowest, and average GSD for each architecture is shown in Table 10.

Table 10. Resolution Results (Meters GSD)

	Traditional	CubeSat
Minimum Value	🟩 0.31	🟨 4.11
Maximum Value	🟨 2.51	🟥 32.74
Mean	🟨 1.78	🟥 23.21
Standard Deviation	0.63	8.26
% of Images Meeting Threshold	100%	10.21%
% of Images Meeting Objective	14.44%	0%

Color Key

🟩	Meets Objective	🟨	Meets Threshold	🟥	Does Not Meet Threshold

Both architectures are capable of meeting the threshold GSD requirement of 10 meters, indicating that both architectures would provide at least some useful imagery in response to a hurricane disaster scenario. The objective requirement of 1 meter GSD is more challenging; only the traditional architecture is capable of meeting this value, and only meets this value 14.44% of the time. A visual comparison of the best, worst, and average resolution of both architectures is provided in Figure 10.

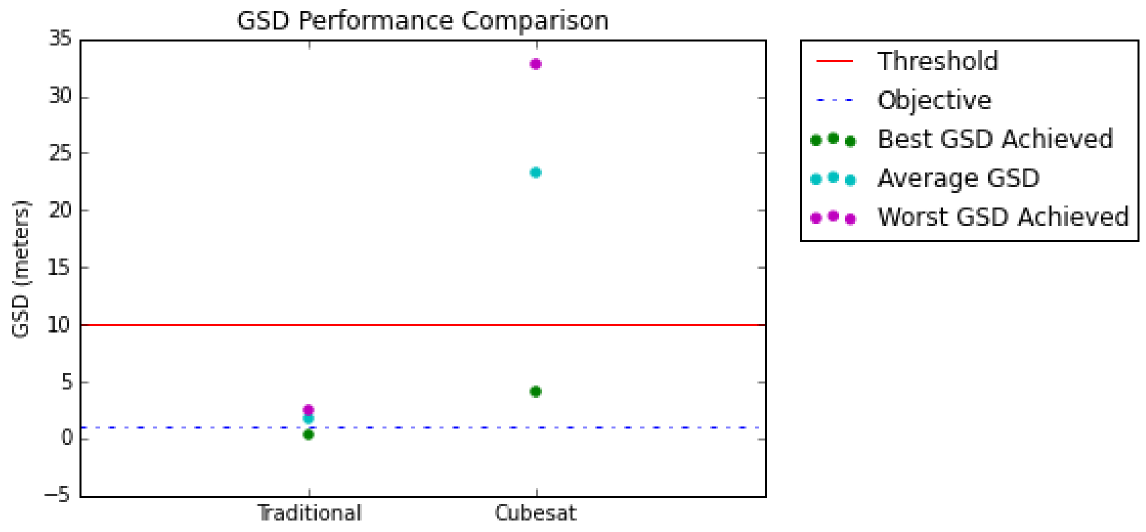


Figure 10. GSD Performance Comparison

Compared to the traditional architecture, the CubeSat architecture has a much higher range of GSD values. As both the average value and Figure 11 show, much of this range is above the 10-meter threshold; most collects from the CubeSat design do not meet resolution requirements. In this simulation, 10.21% of collects meet the threshold requirement, and 0% meet the objective.

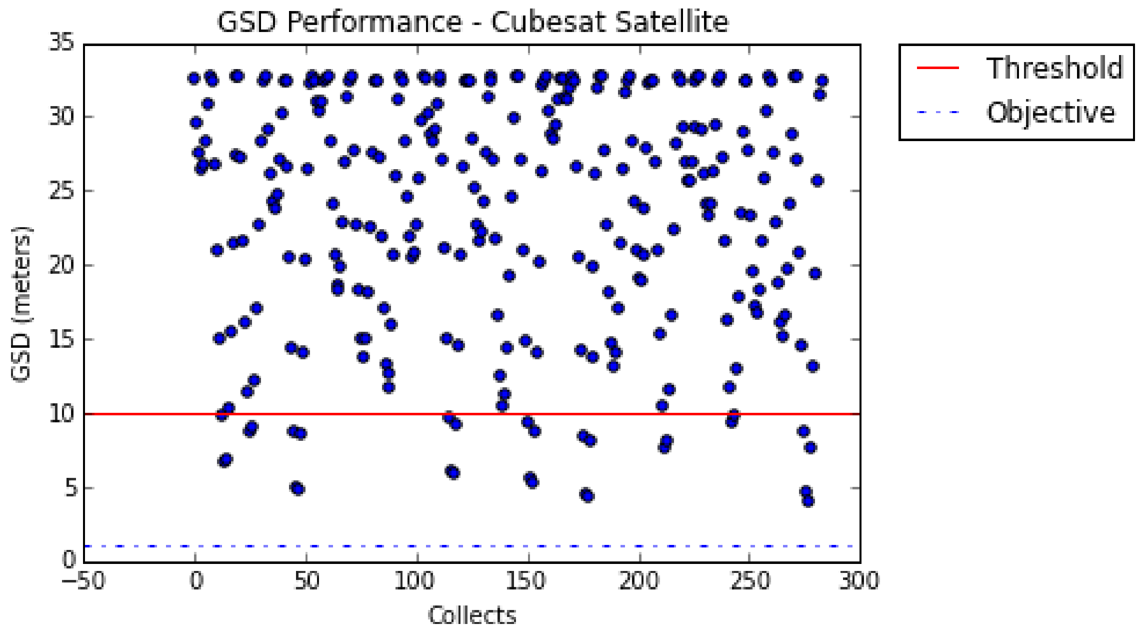


Figure 11. GSD of Individual Collects for a CubeSat Design

As discussed in Chapter 3 and in the next sections, the fact that most CubeSat collects would not meet the threshold requirement is accounted for when calculating timeliness and coverage.

Timeliness

Timeliness is measured from the time of the event to the time the first collect is downlinked to a ground station. The minimum, maximum, and average timeliness, computed from all ascending nodes, for both architectures, is shown in Table 11.

Table 11. Timeliness Results (Hours)

	Traditional	CubeSat
Minimum Value	10.73	10.73
Maximum Value	21.78	23.34
Mean	15.44	16.59
Standard Deviation	3.28	3.50

Color Key

🟩	Meets Objective	🟨	Meets Threshold	🟥	Does Not Meet Threshold

Both architectures meet the 24-hour timeliness objective requirement, meaning that both architectures can provide imagery on a timeline relevant to users. Performance of both architectures is similar, with the average values being within 8% of each other. A visual comparison of these values is provided in Figure 12.

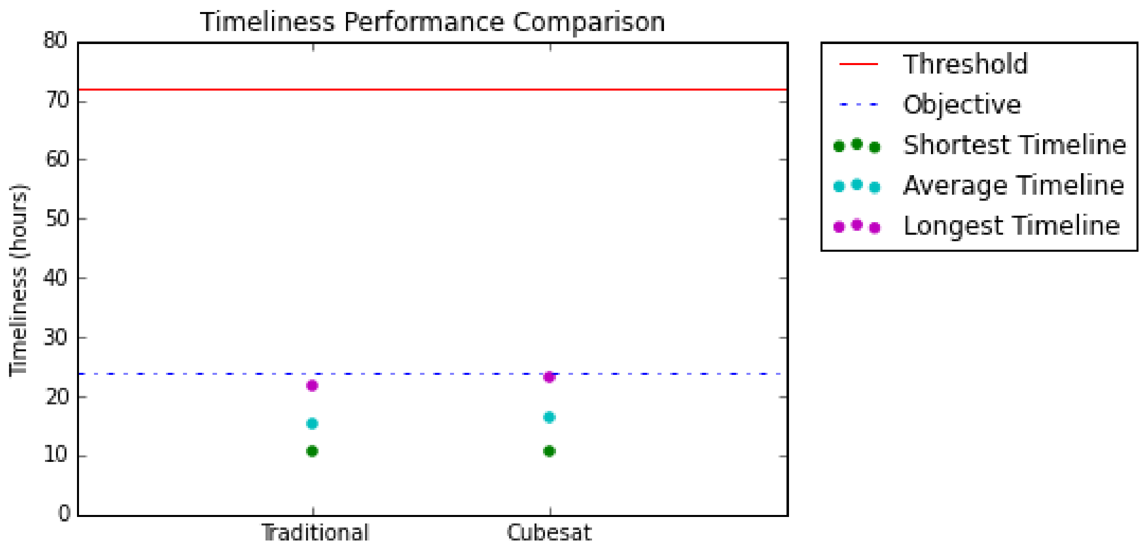


Figure 12. Timeliness Performance Comparison: Maximum, Minimum and Mean Across 360 Ascending Nodes

Coverage

The minimum, maximum, and average coverage, computed using results from all ascending nodes, for both architectures is shown in Table 12.

Table 12. Coverage Results (Sq. Km)

	Traditional	CubeSat
Minimum Value	30938	16692
Maximum Value	46751	38601
Mean	35451	19697
Standard Deviation	3373	3580

Color Key

🟩	Meets Objective	🟨	Meets Threshold	🟥	Does Not Meet Threshold

Both architectures meet the coverage requirement of 11522 square km within 72 hours of the disaster event, with some margin; this is visible in Figure 13, where the solid line is the requirement and the lowest dots are the minimum coverage values achieved.

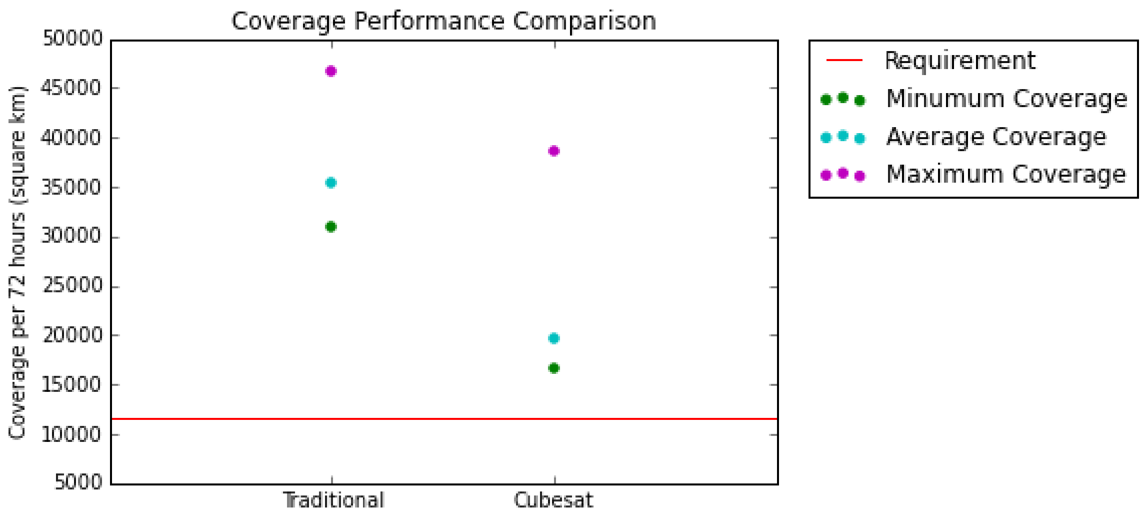


Figure 13. Coverage Performance Comparison: Maximum, Minimum and Mean Across 360 Ascending Nodes

The results for coverage show that both architectures are capable of providing users with images of any location in this given affected area on a relevant timeline. Recall that only collects meeting the threshold GSD requirement were counted towards meeting this requirement; this means that for the CubeSat architecture, even with only 10% of geometries/accesses yielding usable imagery, given enough satellites this design solution is viable. Once again, however, the traditional architecture has more favorable performance, owing to a much higher percentage of images meeting threshold GSD requirements.

Question 2: Results of Cost Modeling

The systems engineering costs for both architectures as computed by the COSYSMO cost model is given in Table 13.

Table 13. COSYSMO Cost Results

	Traditional	CubeSat
Cost ($)	1,163,929	1,117,582
Effort (person-mo.)	116.4	111.8
Schedule (months)	7.2	7.1

The cost results for the CubeSat and traditional architectures are closer than one might initially expect, differing by $46,347, or 3.98%. This similarity reflects three things. First, that only systems engineering effort costs are considered by COSYSMO; if other costs such as detailed design effort, raw materials, manufacturing labor, or launch costs had been investigated, this comparison would likely yield different results.

Second, these numbers reflect the parameters that COSYSMO does and does not model, and the assumptions behind those parameters. For example, satellite size, which is a key parameter by which the architectures differ, is not an input to COSYSMO. For this thesis, there was an assumption that only the functional size parameter in equation (4) varied, with the scale factors and effort multipliers remaining static between architectures; this assumption may need revisiting.

Third, this result reflects the similarity of the two architectures at the level of fidelity modeled for this thesis. With a few exceptions noted in chapter 3, the architectures perform the same system-level functions in order to achieve the same mission requirements. At the level of fidelity modeled, the physical implementation of the C2 and image processing segments also do not vary.

The cost estimate results obtained here, along with these three considerations, indicate that COSYSMO in its current form is not yet a valid means of comparing systems engineering costs of dissimilar architectures. Suggestions for improving COSYSMO for this purpose are discussed in the Conclusions and Recommendations section.

Question 3: Implications of Using MBSE

This thesis addressed two related but distinct questions: how do two architectures compare in terms of performance, and how do these same architectures compare in terms of cost? Answering either question does not strictly require the use of MBSE; question one could be answered with a document of desired design parameters and STK alone, and question two could be attempted with sufficient knowledge of desired design parameters and existing cost models such as USCM. However, the MBSE approach added a level of rigor and understanding that the simpler approach described in the previous sentence would not provide.

As described in Chapter Three, the OOSEM approach to MBSE is scenario based. Identifying and developing a scenario from which to derive requirements ensured the design solutions developed from those requirements were practical, making for a more realistic comparison. Using MBSE and SysML to capture the derived requirements enabled a clearer understanding of those requirements. By ensuring constraint requirements were captured and understood, the quantitative trade spaces became better defined.

The initial development of a SysML architecture model was time consuming; however, once a baseline architecture was established, it was relatively easy to modify. This was especially noted in the development of the traditional and CubeSat SysML models. The traditional model was built first; this process took several weeks[10]. The CubeSat model was developed from the baseline of the traditional architecture model, with necessary changes to requirements, physical parameters/constraints, etc. This step took days rather than weeks.

By virtue of being a systems tool, rather than a domain-focused tool, CAMEO enabled the first two investigative questions to extend beyond the satellite designs themselves. Ground station placement and performance were key variables for the timeliness and coverage MOEs; model views that included the C2 and image processing segments were relevant inputs to the cost model. The usage of a generic systems method and tool ensured the entire system of interest, along with relevant external elements such as AFSCN, could be modeled and accounted for.

The converse to this is the risk of a model becoming too generic, when some degree of domain-specific focus is required. For this thesis, the CubeSat SysML system model was supplied to the COSYSMO system cost modeling tool; an alternate approach could have involved having a CubeSat CAD physical model supply physical parameters to a satellite-specific cost model. Determining which approach provides a better cost estimate requires further research.

[10] Though some of this timeline is attributable to a learning curve associated with CAMEO.

Summary

This section described the quantitative results of STK performance analysis and of COSYSMO cost estimation for both the traditional and CubeSat architectures, and discussed these results in the context of the first two investigative questions. A discussion on using MBSE methods as part of the analysis process answered the third investigative question.

V. Conclusions and Recommendations

Chapter Overview

This section highlights the conclusions reached from the investigative questions, and discusses new questions uncovered during this research that should be addressed in further research.

Conclusions of Research

While it is intuitive that CubeSat-sized satellites would not directly match the performance of a larger traditional satellite architecture, this thesis demonstrated that the utility of CubeSats is not all that diminished compared to traditional architectures. Ground resolution is the most significant disparity between the two solutions. A CubeSat architecture can provide useful EO imagery in the sub-10-meter range for a portion of collects, but cannot meet a sub-meter requirement; a traditional architecture easily meets a sub-10-meter GSD requirement, and can meet a sub-meter requirement for a portion of collects. In terms of user needs in a disaster scenario, these results mean that CubeSat architecture imagery would be useful for identifying broader phenomena such as areas of flooding, but could not identify features such as individual structure damage. Imagery from the traditional architecture would be useful in addressing all user needs, but higher-resolution imagery would be less frequent.

For coverage, both architectures are capable of providing sub-10-meter GSD imagery covering the entire island within 72 hours. For the traditional architecture, this requires one satellite. The CubeSat architecture requires 3 satellites, owing partially to the fact that only a percentage of CubeSat collects meet the sub-10-meter GSD threshold. For this scenario and set of design solutions, timeliness between architectures is comparable, with timelines meeting user needs for both architectures.

Counter-intuitively, the results of the COSYSMO cost estimates for the two architectures where within 4% of each other. Reasons for this likely include:

1. Cost estimation comparison was limited to system engineering costs
2. Architectures were quite similar at the functional level
3. The physical parameters by which the architectures varied most significantly are not parameters captured by COSYSMO

As such, this research demonstrated that satellite architectures modeled using MBSE can provide input to cost estimation tools such as COSYSMO. However, this approach requires refinement for the purposes of trade studies.

Significance of Research

In recent years, the maturation and proliferation of CubeSat designs have generated interest in their usage operationally. The results of this thesis broadly suggest that, for remote sensing, CubeSats can perform the same mission as a traditional architecture, though with sensor performance limitations. These results are consistent with previous research such as McKenney (2016), and with the achievements of commercial companies such as Planet Labs.

The Model-Based Systems Engineering approach enabled a disciplined method for developing and comparing the two architectures, demonstrating the method’s usefulness in performing similar analysis for other trade studies. This result is consistent with previous research such as Thompson (2015).

Recent research has suggested that SysML models can be integrated into systems engineering cost estimation tools such as COSYSMO (Edwards, 2016). The results of this thesis suggest this approach is not without its limitations in the space domain. This approach merits further investigation to determine how best to address these limitations.

Recommendations for Future Research

This thesis addresses questions specific to two specific architecture implementations. However, throughout its development the intent was that the method and models could be generalized to address any number of related questions, in keeping with the philosophy of MBSE. To this end, Figure 14 displays the approach described in Chapter 3, Figure 1, but with general suggestions on areas for further exploration.

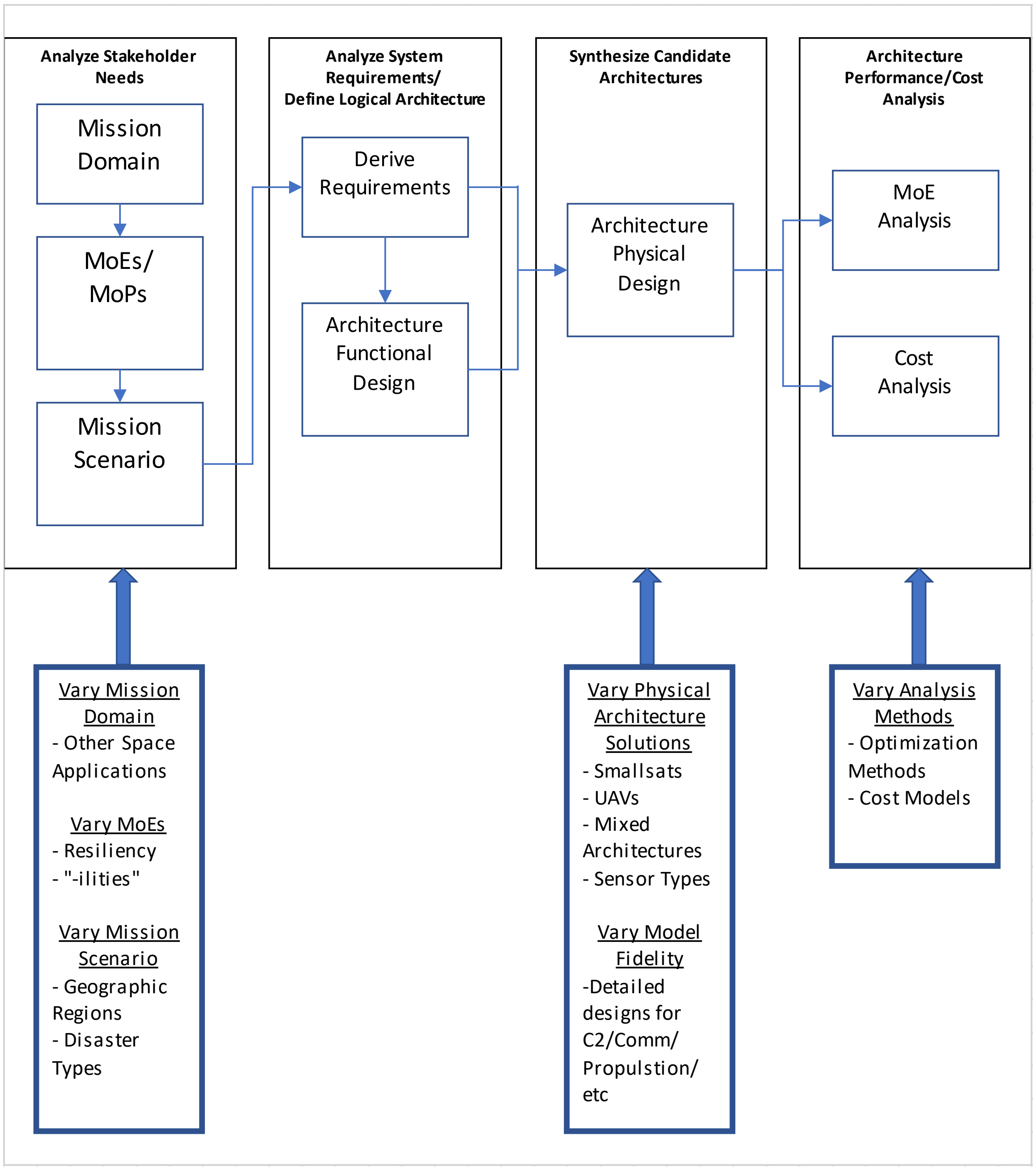

Analyze Stakeholder Needs
Mission Domain
MoEs/ MoPs
Mission Scenario
Analyze System Requirements/ Define Logical Architecture
Derive Requirements
Architecture Functional Design
Synthesize Candidate Architectures
Architecture Physical Design
Architecture Performance/Cost Analysis
MoE Analysis
Cost Analysis
Vary Mission Domain
- Other Space Applications
Vary MoEs
- Resiliency
- "-ilities"
Vary Mission Scenario
- Geographic Regions
- Disaster Types
Vary Physical Architecture Solutions
- Smallsats
- UAVs
- Mixed Architectures
- Sensor Types
Vary Model Fidelity
-Detailed designs for C2/Comm/ Propulstion/ etc
Vary Analysis Methods
- Optimization Methods
- Cost Models

Figure 14. Incorporating Future Research into Existing Thesis Methodology

A follow-on trade study incorporating an MOE for resiliency would be of particular interest; besides cost, resiliency may be another area in which CubeSats have advantages compared to traditional architectures. In addition to an investigation of resiliency trade studies, there are any number of ways this method and these models could be used and modified to investigate related areas. Investigation of CubeSat utility in other space-related applications, or against other real-world scenarios, could further validate the results of this thesis. The incorporation of UAV or other remote sensing platform models would provide both further validation of this methodology and practical results for further system development. The lists in Figure 14 are not exhaustive, but are meant to spark ideas that future students could use for their own research.

Integration of MBSE models with COSYSMO for the purposes of spacecraft system engineering cost modeling is an area for significant future research. A starting point would be to compare these results to the System Engineering and Program Management CER in the USCM. This CER is a function of spacecraft bus weight, payload weight, and an integration and test parameter (Space and Missile Systems Center, 2015). A comparison would have to be limited to the spacecraft-specific portions of the architectures developed in this thesis, as the USCM does not incorporate ground C2 or data processing components.

Following that should be a more in-depth investigation and calibration of all relevant parameters in the COSYSMO cost model. This thesis looked only at functional size; it is possible that other parameters such as calibration factors, scale factors, and effort multipliers could significantly affect results. Finally, the COSYSMO model itself should be updated to better reflect space domain-unique aspects affecting systems engineering cost. The relationship between the size or weight of a spacecraft and its systems engineering costs as a percentage of overall program costs should specifically be investigated, as this would incorporate the relevant parameters of both COSYSMO and spacecraft cost models such as USCM.

Summary

This section briefly summarized the primary conclusions derived from this research. A discussion of the significance of the research demonstrate where it validated previous research, and where more research is required. Ideas for future research included similar trade studies with different MOEs, investigating trades on a wider variety of platforms, and further exploration of using MBSE models with COSYSMO for the purposes of spacecraft system engineering cost modeling.

Appendix A. Use Cases for Traditional Architecture

Conduct Imagery Operations

Preconditions: 1) Satellite architecture on-orbit, checked out and in good health (including ground stations/processing). 2) Target list has been established and is ready for TA to input into system.

Post Conditions: 1) Satellite imagery has been collected, processed, and made available to outside database and/or imagery analyst.

Assumptions: 1) both architectures are dedicated to this specific disaster, with no competing collects in the region. 2) both architectures will collect and forward data to next available ground contact: no on-orbit relay. 3) Tasking strategy: Both architectures are taskable and steerable (i.e. not just staring and collecting on open-loop tasking) 4) Cloud cover is not prohibiting collects pre-and post-disaster 5) Weather data is not specifically being provided – dedicated weather satellites outside the system boundary fulfill this need. 6) Satellite(s) would not need to maneuver and expend fuel for better access to areas affected by individual disasters.

Actors: Imagery Analyst (IA), Tasking Authority (TA)

Use case: Tasking Authority inputs target list into Ground/C2 subsystem. Ground/C2 subsystem returns acknowledgement of input. Ground/C2 processes target list into an executable imaging schedule. Ground/C2 uplinks image schedule to satellite(s) during earliest available comm window(s). Upon having access to a target in the image schedule, satellite images target and stores image data in on-board storage. Satellite repeats this sequence until a comm window with a ground station opens. Satellite stops imaging (if need be) and downlinks image data to ground station. Image data is processed and made available to IA. Satellite continues to execute against image schedule until a new image schedule is received.

Maneuver satellite

Preconditions: known initial orbit; known final orbit; known stationkeeping/maneuver data

Post Conditions: satellite achieves desired orbital parameters; system ready to execute "Conduct Imagery Operations" use case.

Assumptions: System is not currently being tasked against active disasters

Actors: Satellite operator (SO), Tasking Authority (TA)

Satellite operator inputs relevant stationkeeping/maneuver data into Ground/C2 subsystem, including command to cease imaging. Ground/C2 makes TA aware of planned non-availability of satellite for imaging. Ground/C2 uplinks commands to satellite(s) during earliest available comm window(s). Satellite executes burn maneuver(s). Satellite continuously sends back telemetry via omni antenna/AFSCN. Satellite operator monitors telemetry for anomalies. Ground/C2 makes TA aware that satellite is available for imaging. "Conduct Imagery Operations" use case resumes.

<u>Task system</u>

Preconditions: 1) Satellite architecture on-orbit, checked out and in good health (including ground stations/processing). 2) Target list has been established and is ready for TA to input into system.

Post Conditions: 1) Satellite has received imaging commands and is ready to begin imaging

Assumptions: target list is regional; tasking list is pre-organized by priority (first target on list is highest priority); system does not optimize collection (first target on list is first target imaged, and so on until list is complete); translation of tasking list into spacecraft imaging commands happens on the ground.

Use case: Tasking Authority inputs target list into a Mission Tasking Interface (MTI) (part of Ground/C2 subsystem). MTI passes tasking list to processor. Processor identifies upcoming image windows for targets on tasking list. Processor identifies which satellite has the window and length of window, and assigns number of targets per window based on estimated imaging time. Processor continues until all targets are assigned imaging windows in an “image schedule”. Processor translates image schedule into set of commands for each individual satellite. Processor sends commands to a buffer at ground antenna for uplink at next available comm window. Ground antenna uplinks commands during comm window. Satellite sends acknowledgement signal. Satellite tracks time until next image window.

<u>Image targets</u>

Preconditions: 1) Satellite has received imaging commands and is ready to begin imaging

Post Conditions: 2) Satellite has imaged all targets assigned

<u>Assumptions:</u> target list is regional, meaning an image window consists of access to multiple targets with close proximity; tasking list is pre-organized by priority (first target on list is highest priority); system does not optimize collection (first target on list is first target imaged, and so on until list is complete); translation of tasking list into spacecraft imaging commands happens on the ground; on-board storage is sufficient to hold all imaging collects between comm windows.

<u>Use Case:</u> Satellite is on standby until near target access/image window. Just outside of target access, satellite slews to point at the first target. Upon reaching target access, satellite payload images target. Image data is sent from payload to on-board storage buffer. Satellite slews slightly to next target and repeats imaging. Satellite continues slewing/imaging until end of image window. Satellite returns to standby mode until next imaging or comm window.

<u>Communicate with satellite</u>

Preconditions: 1) Data is ready to be exchanged between ground antenna and satellite. 2)

Post conditions: 1) Data has been exchanged between ground antenna and satellite.

Assumptions: Ground station knows/reasonably predicts where satellite is; satellite does not know where ground station is. Data exchange is of sufficiently short duration such that it will fit inside comm window.

Satellite is on standby until near antenna access/comm window. Ground station recognizes that satellite is entering comm window, and sends a message to satellite's omni-directional antenna with commands to slew main antenna to point at ground antenna. Satellite slews so that main comm antenna points at ground antenna. Ground antenna uplinks commands. Satellite sends acknowledgement signal. Satellite downloads image data, telemetry data. Ground antenna passes mission data to image processing, and telemetry data to Ground/C2 processor.

Once all data has been exchanged, satellite returns to standby mode.

Process imagery

Preconditions: 1) Data has been downlinked to ground antenna

Post conditions: 1) A full processed image has been delivered/disseminated/made available to imagery analysts

Assumptions: Image retrieval/display for imagery analysts is outside of system boundary.

Use Case: Mission Data is sent to Image Processor from Ground Antenna. Image Processor ingests Mission Data and performs functions to form a softcopy image. If pre-event imagery of tasked targets is available, the Image Processor also generates a change detection product. Image Processor passes softcopy image/change detection product to Imagery Database for access by Imagery Analysts. A notification is sent to subscribers informing them of which targets have imagery now available.

Troubleshoot Spacecraft Anomaly

Preconditions: Spacecraft has encountered an anomaly

Post conditions: Spacecraft has recovered from anomaly and returned to operations.

Assumptions: Spacecraft anomaly is recoverable (i.e. it didn't explode); spacecraft is not able to communicate via normal comm link with ground station (main comm failure/nav failure/other reasons)

Use Case: Upon encountering an anomaly, the spacecraft goes into a "safe mode". Spacecraft broadcasts "safemode" telemetry (error codes) via omni antenna to AFSCN network. AFSCN network relays telemetry to C2 processor, on to Spacecraft Operator. Operator begins running troubleshooting checklist. C2 processor automatically drops sick bird from imaging/comm schedules. Operator sends commands to solve anomaly. C2 processor routes commands through AFSCN to spacecraft. Spacecraft receives and executes commands to recover. Spacecraft acknowledges recovery via AFSCN; C2 processor adds bird back into imaging/comm schedule. "Conduct Imagery Operations" use case resumes.

<extend> Communicate via AFSCN

Spacecraft continuously broadcasts “safemode” telemetry (error codes) via omni antenna. Upon satellite coming within range of AFSCN ground site, signal is presumably received by AFSCN. AFSCN relays to C2 processor, and awaits response. Once C2 segment begins anomaly troubleshooting, AFSCN continues providing a relay, scheduling AFSCN comm windows and managing message traffic as necessary. Once satellite has recovered and acknowledged recovery, C2 segment discontinues using AFSCN to communicate with satellite.

Appendix B. Use Cases for CubeSat Architecture

Conduct Imagery Operations

Preconditions: 1) Satellite architecture on-orbit, checked out and in good health (including ground stations/processing). 2) Target list has been established and is ready for TA to input into system.

Post Conditions: 1) Satellite imagery has been collected, processed, and made available to outside database and/or imagery analyst.

Assumptions: 1) both architectures are dedicated to this specific disaster, with no competing collects in the region. 2) both architectures will collect and forward data to next available ground contact: no on-orbit relay. 3) Tasking strategy: Both architectures are taskable and steerable (i.e. not just staring and collecting on open-loop tasking) 4) Cloud cover is not prohibiting collects pre-and post-disaster 5) Imagery is not for purposes of weather monitoring – GOES/POES/etc fulfill this need. 6) Satellite(s) would not need to maneuver and expend fuel for better access to areas affected by individual disasters.

Actors: Imagery Analyst (IA), Tasking Authority (TA)

Use case: Tasking Authority inputs target list into Ground/C2 subsystem. Ground/C2 subsystem returns acknowledgement of input. Ground/C2 processes target list into an executable imaging schedule. Ground/C2 uplinks image schedule to satellite(s) during earliest available comm window(s). Upon having access to a target in the image schedule, satellite images target and stores image data in on-board storage. Satellite repeats this sequence until a comm window with a ground station opens. Satellite stops imaging (if need be) and downlinks image data to ground station. Image data is processed and made available to IA. Satellite continues to execute against image schedule until a new image schedule is received.

Note: while alternative CONOPS/use cases for cubesats are possible, it was a design decision to keep this Use Case static between the two architectures to better meet mission requirements.

Task system

Preconditions: 1) Satellite architecture on-orbit, checked out and in good health (including ground stations/processing). 2) Target list has been established and is ready for TA to input into system.

Post Conditions: 1) Satellite has received imaging commands and is ready to begin imaging

Assumptions: target list is regional; tasking list is pre-organized by priority (first target on list is highest priority); system does not optimize collection (first target on list is first target imaged, and so on until list is complete); translation of tasking list into spacecraft imaging commands happens on the ground, *spacecraft can take multiple images per imaging window (IW).*

Use case: Tasking Authority inputs target list into a Mission Tasking Interface (MTI) (part of Ground/C2 subsystem). MTI passes tasking list to processor. Processor identifies upcoming image windows for targets on tasking list. Processor identifies which satellite has the window and length of window, and assigns number of targets per window based on estimated imaging time. *Processor predicts GSD for each image, and only schedules a collect if predicted image resolution meets threshold GSD.* Processor continues until all targets are assigned imaging windows in an "image schedule". Processor translates image schedule into set of commands for each individual satellite. Processor sends commands to a buffer at ground antenna for uplink at next available comm window. Ground antenna uplinks commands during comm window. Satellite sends acknowledgement signal. Satellite tracks time until next image window.

Image targets

Preconditions: 1) Satellite has received imaging commands and is ready to begin imaging

Post Conditions: 2) Satellite has imaged all targets assigned

Assumptions: target list is regional, meaning an image window consists of access to multiple targets with close proximity; tasking list is pre-organized by priority (first target on list is highest priority); system does not optimize collection (first target on list is first target imaged, and so on until list is complete); translation of tasking list into spacecraft imaging commands happens on the ground; on-board storage is sufficient to hold all imaging collects between comm windows.

Use Case: Satellite is on standby until near target access/image window. Just outside of target access, satellite slews to point at the first target. Upon reaching target access, satellite payload images target. Image data is sent from payload to on-board storage buffer. Satellite slews slightly to next target and repeats imaging. Satellite continues slewing/imaging until end of image window. Satellite returns to standby mode until next imaging or comm window.

Communicate with satellite

Preconditions: 1) Data is ready to be exchanged between ground antenna and satellite. 2)

Post conditions: 1) Data has been exchanged between ground antenna and satellite.

Assumptions: Ground station knows/reasonably predicts where satellite is; satellite does not know where ground station is. Data exchange is of sufficiently short duration such that it will fit inside comm window.

Satellite is on standby until near antenna access/comm window. Ground station recognizes that satellite is entering comm window and slews ground antenna to make contact. Satellite slews so that main comm antenna points at ground antenna. Ground antenna uplinks commands. Satellite sends acknowledgement signal. Satellite downloads image data, telemetry data. Ground antenna passes mission data to image processing, and telemetry data to Ground/C2 processor.

Once all data has been exchanged, satellite returns to standby mode.

Process imagery

Preconditions: 1) Data has been downlinked to ground antenna

Post conditions: 1) A full processed image has been delivered/disseminated/made available to imagery analysts

Assumptions: Image retrieval/display for imagery analysts is outside of system boundary.

Use Case: Mission Data is sent to Image Processor from Ground Antenna. Image Processor ingests Mission Data and performs functions to form a softcopy image. If pre-event imagery of tasked targets is available, the Image Processor also generates a change detection product. Image Processor passes softcopy image/change detection product to Imagery Database for access by Imagery Analysts. A notification is sent to subscribers informing them of which targets have imagery now available.

<u>Troubleshoot Spacecraft Anomaly (modified for Cubesat)</u>

Preconditions: Spacecraft has encountered an anomaly

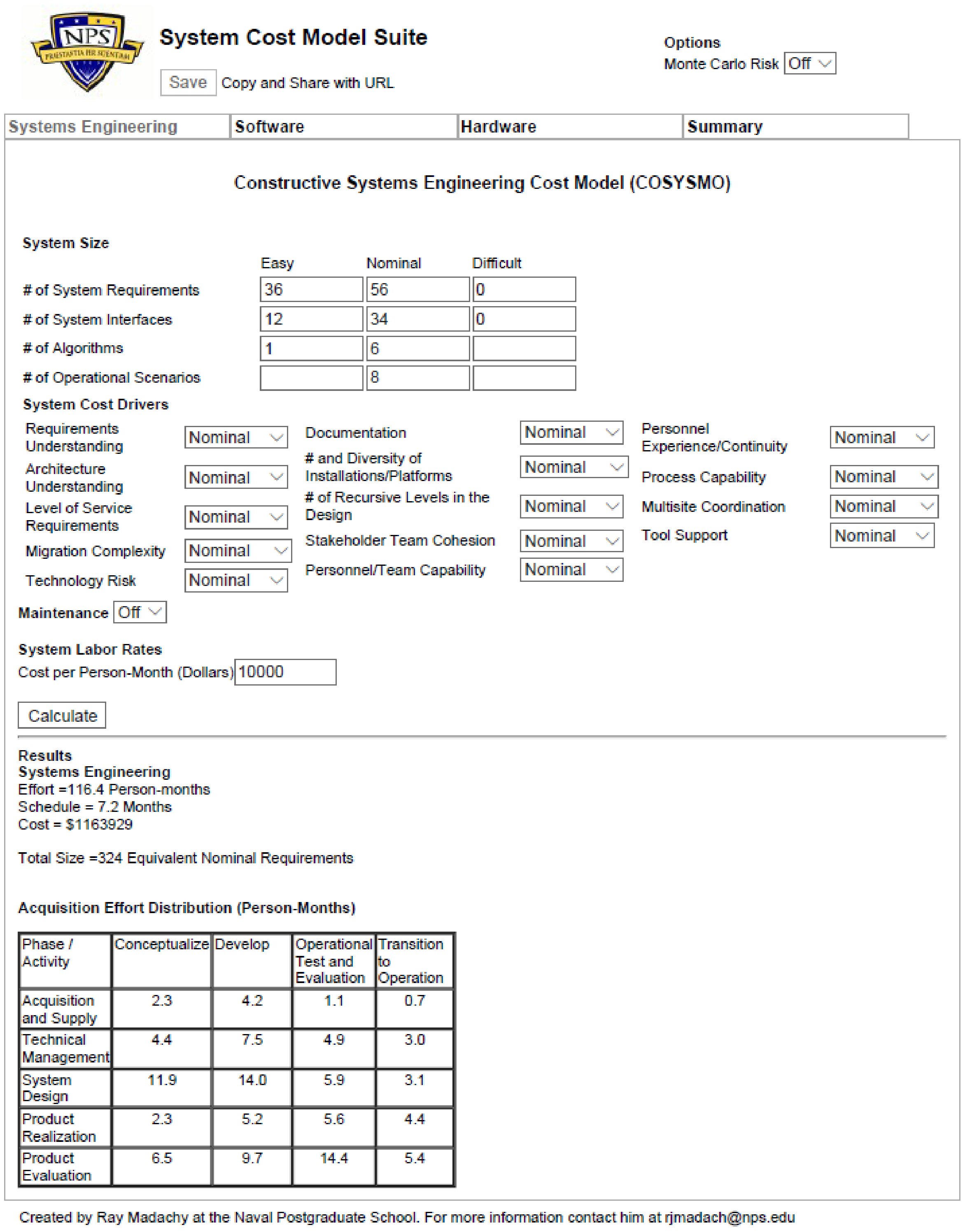
Post conditions: Spacecraft has recovered from anomaly and returned to operations.

Assumptions: Spacecraft anomaly is recoverable (i.e. it didn't explode, comm still works)

Use Case: Upon encountering an anomaly, the spacecraft goes into a "safe mode". Spacecraft broadcasts "safe mode" telemetry (error codes) via comm antenna. Receiving ground antenna relays telemetry to C2 processor, on to Spacecraft Operator. C2 processor automatically drops sick bird from imaging/comm schedules. Spacecraft executes a recovery algorithm to attempt recovery. C2 processor attempts to communicate with bird at every comm window. Once spacecraft is recovered and contact is re-established, C2 processor adds bird back into imaging/comm schedule. "Conduct Imagery Operations" use case resumes.

<extend> If satellite fails to recovery automatically, Operator begins running troubleshooting checklist. Operator sends commands to solve anomaly. C2 processor routes commands to spacecraft. Spacecraft receives and executes commands to recover. Original Use Case continues.

Appendix C. NPS Cost Model Suite COSYSMO Output: Traditional Architecture



System Cost Model Suite

Save Copy and Share with URL

Options

Monte Carlo Risk Off

Systems Engineering | Software | Hardware | Summary

Constructive Systems Engineering Cost Model (COSYSMO)

System Size

	Easy	Nominal	Difficult
# of System Requirements	36	56	0
# of System Interfaces	12	34	0
# of Algorithms	1	6	
# of Operational Scenarios		8	

System Cost Drivers

Requirements Understanding	Nominal	Documentation	Nominal	Personnel Experience/Continuity	Nominal
Architecture Understanding	Nominal	# and Diversity of Installations/Platforms	Nominal	Process Capability	Nominal
Level of Service Requirements	Nominal	# of Recursive Levels in the Design	Nominal	Multisite Coordination	Nominal
Migration Complexity	Nominal	Stakeholder Team Cohesion	Nominal	Tool Support	Nominal
Technology Risk	Nominal	Personnel/Team Capability	Nominal		

Maintenance Off

System Labor Rates

Cost per Person-Month (Dollars) 10000

Calculate

Results
Systems Engineering
Effort =116.4 Person-months
Schedule = 7.2 Months
Cost = $1163929

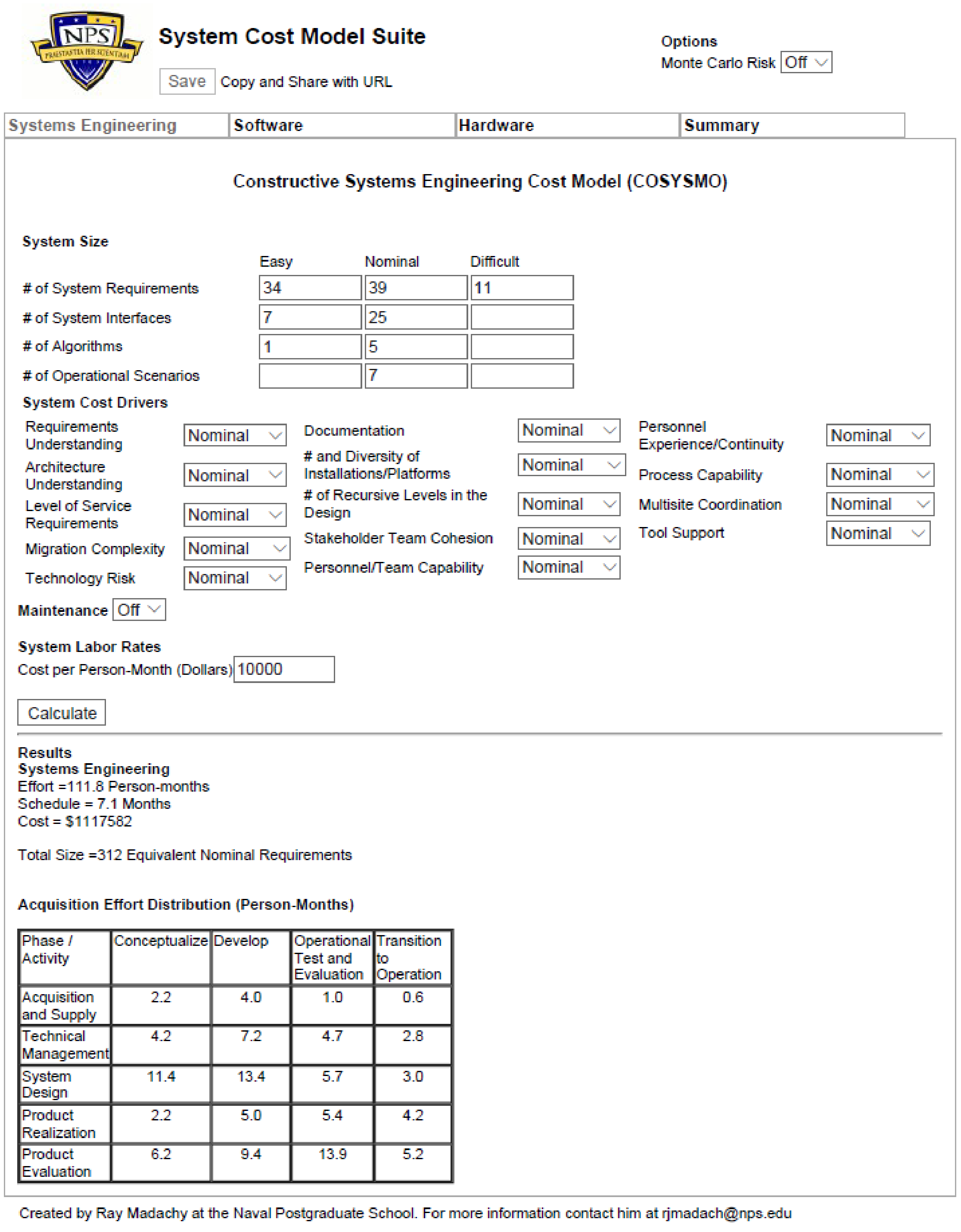
Total Size =324 Equivalent Nominal Requirements

Acquisition Effort Distribution (Person-Months)

Phase / Activity	Conceptualize	Develop	Operational Test and Evaluation	Transition to Operation
Acquisition and Supply	2.3	4.2	1.1	0.7
Technical Management	4.4	7.5	4.9	3.0
System Design	11.9	14.0	5.9	3.1
Product Realization	2.3	5.2	5.6	4.4
Product Evaluation	6.5	9.7	14.4	5.4

Created by Ray Madachy at the Naval Postgraduate School. For more information contact him at rjmadach@nps.edu

Appendix D. NPS Cost Model Suite COSYSMO Output: CubeSat Architecture



System Cost Model Suite

Save Copy and Share with URL

Options
Monte Carlo Risk: Off

Systems Engineering | Software | Hardware | Summary

Constructive Systems Engineering Cost Model (COSYSMO)

System Size

	Easy	Nominal	Difficult
# of System Requirements	34	39	11
# of System Interfaces	7	25	
# of Algorithms	1	5	
# of Operational Scenarios		7	

System Cost Drivers

Requirements Understanding	Nominal	Documentation	Nominal	Personnel Experience/Continuity	Nominal
Architecture Understanding	Nominal	# and Diversity of Installations/Platforms	Nominal	Process Capability	Nominal
Level of Service Requirements	Nominal	# of Recursive Levels in the Design	Nominal	Multisite Coordination	Nominal
Migration Complexity	Nominal	Stakeholder Team Cohesion	Nominal	Tool Support	Nominal
Technology Risk	Nominal	Personnel/Team Capability	Nominal		

Maintenance Off

System Labor Rates
Cost per Person-Month (Dollars) 10000

Calculate

Results
Systems Engineering
Effort =111.8 Person-months
Schedule = 7.1 Months
Cost = $1117582

Total Size =312 Equivalent Nominal Requirements

Acquisition Effort Distribution (Person-Months)

Phase / Activity	Conceptualize	Develop	Operational Test and Evaluation	Transition to Operation
Acquisition and Supply	2.2	4.0	1.0	0.6
Technical Management	4.2	7.2	4.7	2.8
System Design	11.4	13.4	5.7	3.0
Product Realization	2.2	5.0	5.4	4.2
Product Evaluation	6.2	9.4	13.9	5.2

Created by Ray Madachy at the Naval Postgraduate School. For more information contact him at rjmadach@nps.edu

Bibliography

Analytical Graphics Incorporated. (2017, March). *Sensor Resolution*. Retrieved from STK Programming Help: help.agi.com/stk/11.1/#stk/sn-13.htm?Highlight=GSD

Battersby, S. E., Hodgson, M. E., & Wang, J. (2012). Spatial Resolution Imagery Requirements for Identifying Structure Damage in a Hurricane Disaster: A Cognitive Approach. *Photogrammetric Engineering & Remote Sensing*, 625-635.

Boshuizen, C. R., Mason, J., Klupar, P., & Spanhake, S. (2014). Results from the Planet Labs Flock Constellation. *28th Annual AIAA/USU Conference on Small Satellites.* AIAA.

Central Intelligence Agency. (2017, November 14). *Puerto Rico*. Retrieved from The World Factbook: https://www.cia.gov/library/publications/the-world-factbook/geos/rq.html

Delligatti, L. (2014). *SysML Distilled.* Pearson Education.

Department of Homeland Security. (2013, February 22). Remote Sensing and Incident Support. 32. Retrieved from www.napsgfoundation.org/wp-content/uploads/2013/02/NAPSG-Remote-Sensing-Webcast-022213.pdf

DigitalGlobe. (2016, October 7). *System-Ready Imagery.* Retrieved from DigitalGlobe Web site: https://dg-cms-uploads-production.s3.amazonaws.com/uploads/document/file/27/BasicImagery_DS_10-7-16.pdf

Edwards, D. J. (2016). *Exploring the Integration of COSYSMO with a Model-Based Systems Engineering Methodology in Early Tradespace Analytics and Decisions.* Masters Thesis, Naval Postgraduate School, Monterey.

European Space Agency. (2017). *WorldView-2*. Retrieved from eoPortal Directory Website: https://directory.eoportal.org/web/eoportal/satellite-missions/v-w-x-y-z/worldview-2

European Space Agency. (n.d.). *CubeSat Concept*. Retrieved from eoPortal Directory: https://directory.eoportal.org/web/satellite-missions/c-missions/cubesat-concept

Evans, H., Lange, J., & Schmitz, J. (2014). *The Phenomenology of Intelligence-focused Remote Sensing.* New York: Riverside Research.

Friedenthal, S., Moore, A., & Steiner, R. (2015). *A Practical Guide to SysML, 3rd.* Waltham: Elsevier.

Google. (n.d.). [Geographic Center of Puerto Rico]. Retrieved from www.google.com/maps

Hodgson, M. E., Davis, B. A., Cheng, Y., & Miller, J. (2010). Modeling Remote Sensing Satellite Collection Opportunity Likelihood for Hurricane Disaster Response. *Cartography and Geographic Information Science*, 7-15.

Hoque, M. A.-A., Phinn, S., Roelfsema, C., & Childs, I. (2017). Tropical Cyclone Disaster Management Using Remote Sensing and Spatial Analysis: A Review. *International Journal of Disaster Risk Reduction*, 345-354.

Kaslow, D., Soremuken, G., Hongman, K., & Spangelo, S. (2014). Integrated Model-Based Systems Engineering (MBSE) Applied to the Simulation of a CubeSat Mission. *IEEE Aerospace Conference*, (pp. 1-14). Big Sky, MT.

Konecny, G. (2004). Small Satellites - A Tool for Earth Observation? *XXth ISPRS Congress Technical Commission IV*, (pp. 580-582). Istanbul.

Krueger, J. K., Selva, D., Smith, M. W., & Keesee, J. (2009). Spacecraft and Constellation Design for a Continuous Responsive Imaging System in Space. *AIAA SPACE 2009 Conference & Exposition.* Pasadena: AIAA.

Lillesand, T. M., Kiefer, R. W., & Chipman, J. W. (2008). *Remote Sensing and Image Interpretation, 6th.* John Wiley & Sons.

McKenney, S. J. (2016). *Meeting the DoD's tactical weather needs using cubesats.* Wright-Patterson AFB: Department of the Air Force Air University.

National Academy of Sciences. (2016). *Achieving Science with Cubesats: Thinking Inside the Box.* Washington DC: The National Academies Press.

Pavalkis, S., Papke, B., & Wang, G. (2017). Enabling Repeatable SE Cost Estimation with COSYSMO and MBSE. *27th Annual INCOSE International Symposium.* Adelaide.

Planet. (2015, September). *Planet Spacecraft Operations and Ground Control Version 1.2*. Retrieved from planet.com: https://www.planet.com/docs/spec-sheets/spacecraft-ops/

Schmidt, S., Achenbach, J., & Somashekhar, S. (2017, September 20). *Puerto Rico entirely without power as Hurricane Maria hammers island with devastating force*. Retrieved from Washingtonpost.com: https://www.washingtonpost.com/news/post-nation/wp/2017/09/20/hurricane-maria-takes-aim-at-puerto-rico-with-force-not-seen-in-modern-history/

Sellers, J. J., Astore, W. J., Giffen, R. B., & Larson, W. J. (2005). *Understanding Space: An Introduction to Astronautics* (3rd ed.). New York: McGraw-Hill.

Selva, D., & Krejci, D. (2012). A Survey and Assessment of the Capabilities of Cubesats for Earth Observation. *Acta Astronautica*, 50-68.

Selva, D., & Krejci, D. (2013). Preliminary Assessment of Performance and Cost of a Cubesat Component of the Earth Science Decadal Survey. *27th Annual AIAA/USU Conference on Small Satellites*, (pp. 1-14).

Smith, M. S. (2012, June 23). *EnhancedView News Not so Rosy for GeoEye*. Retrieved from Space Policy Online: http://www.spacepolicyonline.com/news/enhancedview-news-not-so-rosy-for-geoeye

Space and Missile Systems Center. (2015, June 12). Retrieved from Unmanned Space Vehicle Cost Model Online, version 10: www.uscmonline.com

Thompson, R. E. (2015). *A Methodology for the Optimization of Disaggregated Space System Conceptual Designs.* PhD Dissertation, Air Force Institute of Technology, Department of Systems Engineering and Management, Wright-Patterson AFB.

Valerdi, R. (2005). *The Constructive Systems Engineering Cost Model (COSYSMO).* PhD Dissertation, University of Southern California, Los Angeles.

Wertz, J. R., & Larson, W. J. (2010). *Space Mission Analysis and Design, 3rd.* Hawthorne: Microcosm Press.

Womble, J., Ghosh, S., Adams, B. J., & Friedland, C. J. (2006). *Advanced Damage Detection for Hurrican Katrina: Integrating Remote Sensing and VIEWS Field Reconnaisaance.* Buffalo: MCEER.

REPORT DOCUMENTATION PAGE			*Form Approved* *OMB No. 074-0188*
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to an penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**			
1. REPORT DATE *(DD-MM-YYYY)* 22-03-2018	**2. REPORT TYPE** Master's Thesis		**3. DATES COVERED** *(From – To)* March 2010 – March 2018
TITLE AND SUBTITLE **Comparison of Traditional Versus CubeSat Remote Sensing: A Model-Based Systems Engineering Approach**			**5a. CONTRACT NUMBER**
			5b. GRANT NUMBER
			5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Cipera, Daniel L., Captain, USAF			**5d. PROJECT NUMBER**
			5e. TASK NUMBER
			5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/ENY) 2950 Hobson Way, Building 640 WPAFB OH 45433-8865			**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT-ENV-MS-18-M-187
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Intentionally left blank			**10. SPONSOR/MONITOR'S ACRONYM(S)**
			11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **DISTRUBTION STATEMENT A.** APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.			
13. SUPPLEMENTARY NOTES This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.			
14. ABSTRACT This thesis compares the ability of both traditional and CubeSat remote sensing architectures to fulfill a set of mission requirements for a remote sensing scenario. Mission requirements originating from a hurricane disaster response scenario are developed to derive a set of system requirements. Using a Model-based Systems Engineering approach, these system requirements are used to develop notional traditional and CubeSat architecture models. The technical performance of these architectures is analyzed using Systems Toolkit (STK); the results are compared against Measures of Effectiveness (MOEs) derived from the disaster response scenario. Additionally, systems engineering cost estimates are obtained for each satellite architecture using the Constructive Systems Engineering Cost Model (COSYSMO). The technical and cost comparisons between the traditional and CubeSat architectures are intended to inform future discussions relating to the benefits and limitations of using CubeSats to conduct operational missions.			
15. SUBJECT TERMS CubeSat, Space Systems Architecture, Space Systems Modeling			

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Dr. David Jacques, AFIT/ENV
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-3636, ext 3329 (NOT DSN) (david.jacques@afit.edu)
U	U	U	UU	19	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18